Portrait
Dialogue

Referenzhandbuch

Version 6.0 SP1

Pitney Bowes
Software

© 2013 Pitney Bowes Software Inc. Alle Rechte vorbehalten.

Dieses Dokument kann vertrauliche und eigentumsrechtlich geschützte Informationen enthalten, die Eigentum von Pitney Bowes Inc. bzw. seinen Tochter- und Beteiligungsgesellschaften sind.

Die Portrait Software, das Portrait Software-Logo und das Portrait-Zeichen der Portrait Software sind Warenzeichen der Portrait Software International Limited und dürfen in keiner Weise ohne die vorherige, ausdrückliche schriftliche Genehmigung von Portrait Software International Limited benutzt oder verwertet werden.

Warenzeichen

Die Namen von Firmen und Produkten, Marken, Logos und Zeichen, die hier genannt werden, sind möglicherweise Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer.

Portrait Software-Support

Sollten sie über den Inhalt dieser Dokumentation hinausgehende Hilfe benötigen, können sie in der Wissensbasis auf unserer Website **http://support.portraitsoftware.com** nachsehen und den Links zu ihrem Produkt folgen.

Sie können auch andere Portrait Dokumentationen herunterladen. Falls Sie keinen Benutzernamen und kein Kennwort haben oder sie diese vergessen haben sollten, kontaktieren Sie uns über einen der nachstehenden Kanäle.

Wenn sie Probleme bei der Nutzung, Installation oder Dokumentation von diesem Produkt feststellen, kontaktieren sie uns bitte auf einem der folgenden Wege:

E-Mail: support@portraitsoftware.com

Telefon

- USA/Kanada 1-800-335-3860 (gebührenfrei)
- Übrige Welt +44 800 840 0001

Bei einer Problemmeldung ist es hilfreich, wenn Sie uns Folgendes mitteilen können:

- Der Name der Software-Anwendung
- Die Umstände, unter denen das Problem auftrat
- Welche Fehlermeldungen Sie gegebenenfalls gesehen haben
- Die Version der von Ihnen benutzten Software

Pitney Bowes Software Inc.

January 02, 2014

Inhalt

Kapitel 1: Dialogue Admin....11
Dialogue Admin....12
Dialogue Server-Hosts....12
Datenbankinstanzen....12
Kundendomänen....15
Kategorien....15
Dialogeinrichtung....15
Gruppentypen....15
Operationstypen....16
Dialogstatustypen....16
Ereignistypen....16
Anrufstatustypen....16
Einrichtung für unzustellbare E-Mails....17
Aktivitätseinrichtung....18
Aktivitätstypen....19
Optionen für Aufgabennachverfolgung....19
Aufgabenanzeigeintervalle....19
Aufgabenfelder....19
Kanaltypen....19
Ausgabekanäle....20
Nachrichtentypen....22
Ressourcen....22
SQL-Repository....22
Sekundäre Datenbanken....23
Webdienste....23
Plug-In-Repository....23
Benutzer und Sicherheit....25
Benutzer....25
Benutzergruppen....25

Zugriffsrechte....25
Objektsicherheit....29
Benutzersitzungsprotokoll....31
Allgemeine Verwaltung....32
Parametersammlungen....32
Anwendungssysteme....32
Tabellenwartung....32
Test-Dienstprogramme....32

Kapitel 2: Kundendomänen und -ausdrücke....35
Kundendomänen und -ausdrücke....36
Erste Schritte mit Domänen....36
Schritt 1 – Erstellen der SQL-Hauptgruppe....36
Schritt 2 – Hinzufügen der Domäne....37
Schritt 3 – Hinzufügen der Hauptdatengruppe....37
Schritt 4 – Hinzufügen der Felder....38
Schritt 5 – Bearbeiten von Feldeigenschaften....38
Schritt 6 – Aktivieren der neuen Domäne....38
Schritt 7 – Testen der Domäne in Customer View....39
Schritt 8 – Testen der Domäne im Auswahldesigner....39
Schritt 9 – Weitere Informationen zu Domänen....39
Kundendomänen in Visual Dialogue....40
Kundendomänen in Customer View....40
Details zur Konfiguration einer Kundendomäne....42
Eigenschaften von Kundendomänen....42
Datengruppen....43
Felder....47
Datentypen....51
Suchquellen....51
Kundendomänenüberprüfung....52
Ausführen der Kundendomänenüberprüfung....52
Kundendomänenüberprüfung – Nachrichten....53
Formeln....58
Definitionen der Ausdruckssyntax....59
Ausdrucksoperatoren....61
Ausdrucksdatentypen....63
Ausdrucksfunktionen....63

Kapitel 3: Einrichten von Berichten....79
Berichte....80
Berichtanzeige....81
Berichtsformate....83
Berichtsparameter-XML....84
Berichtsansichtsaktionen....86

Kapitel 4: E-Mail- und Verknüpfungsnachverfolgung....89
Nachverfolgung....90
E-Mail Nachverfolgung....90
Verknüpfungsnachverfolgung....91
E-Mail- und Verknüpfungsnachverfolgung – Details....93
Antwortnachverfolgung....95

Kapitel 5: Inhaltsobjekte....99
Inhaltsobjekte....100
Inhaltsobjekt-URLs....100

Kapitel 6: Veröffentlichte Dateien....103
Veröffentlichte Dateien....104
URLs veröffentlichter Dateien....104

Kapitel 7: Verwaltung unzustellbarer E-Mails....107
Verwaltung unzustellbarer E-Mails....108
Systemparameter unzustellbarer E-Mails....108
Protokollierung unzustellbarer E-Mails....109

Kapitel 8: Importieren und Exportieren von Objekten....111
Exportieren und Importieren....112
Exportieren....112
Importieren....113
Einrichten einer komplementären Umgebung....114

Kapitel 9: URL shortening....117

URL-Kürzung **118**
Datenbankinstanzen **119**
Parameter **119**
Nachrichtentypen **120**

Kapitel 10: Plug-In-API **121**
Plug-In-API **122**
Kundendaten-Plug-Ins **123**
Verzweigungs-Plug-Ins **126**
Ausdrucks-Plug-Ins **127**
Generische Plug-Ins **128**
Nachrichten-Plug-Ins **128**
Ausgangskanal-Plug-Ins **129**
Typen **131**
Verzweigungsparametertypen 131
Fragebogen-Datentypen 137
Schnittstellen **138**
IMHAnswerForm-Schnittstelle 138
IMHAnswerFormList-Schnittstelle 142
IMHBranchDynamicList-Schnittstelle 143
IMHBranchInfo-Schnittstelle 145
IMHBranchParamDefs-Schnittstelle 147
IMHBranchPlugin-Schnittstelle 148
IMHCategory-Schnittstelle 149
IMHCacheItem-Schnittstelle 150
IMHCCCallStatusList-Schnittstelle 151
IMHCCProject-Schnittstelle 151
IMHChannelMessage-Schnittstelle 152
IMHMessageContainer-Schnittstelle 153
IMHCmsIdList-Schnittstelle 154
IMHCreateMessagePlugin-Schnittstelle 155
IMHCreateMessagePlugin2-Schnittstelle 157
IMHCreateMessagePlugin3-Schnittstelle 158
IMHContentItem-Schnittstelle 159
IMHContentItemList-Schnittstelle 161
IMHContentObjectRulePlugin-Schnittstelle 162
IMHContentObjectUtils-Schnittstelle 163
IMHContentRequestParams-Schnittstelle 164

IMHControlParamDefs-Schnittstelle....165
IMHCustomer-Schnittstelle....166
IMHCustomerContainer-Schnittstelle....178
IMHCustomerList-Schnittstelle....183
IMHCustomerSortFieldList-Schnittstelle....184
IMHCustomPluginServices-Schnittstelle....185
IMHDataField-Schnittstelle....186
IMHDataFields-Schnittstelle....187
IMHDataGroupPlugin-Schnittstelle....188
IMHDialog-Schnittstelle....190
IMHDialogUtils-Schnittstelle....192
IMHDialogGroup-Schnittstelle....193
IMHDialogOperation-Schnittstelle....195
IMHDialogServerServices-Schnittstelle....196
IMHDuplicateList-Schnittstelle....205
IMHDynamicValuesList-Schnittstelle....206
IMHEmailBounceCodeList-Schnittstelle....207
IMHExprFunctionDef-Schnittstelle....208
IMHExprFunctionPlugin-Schnittstelle....209
IMHGenericPlugin-Schnittstelle....210
IMHLicenseInfo-Schnittstelle....210
IMHMessage-Schnittstelle....211
IMHMessageAssembleInfo-Schnittstelle....214
IMHMessageAttachment-Schnittstelle....215
IMHMessageAttachmentList-Schnittstelle....216
IMHMessageBundle-Schnittstelle....217
IMHMessageUtils-Schnittstelle....219
IMHOutputChannelInfo-Schnittstelle....220
IMHOutputChannelPlugin-Schnittstelle....221
IMHParticipant-Schnittstelle....222
IMHParticipantContainer-Schnittstelle....223
IMHPlugin-Schnittstelle....227
IMHQryAlternative-Schnittstelle....227
IMHQryQuestion-Schnittstelle....228
IMHQrySection-Schnittstelle....230
IMHQuestionnaire-Schnittstelle....231
IMHQuestionnaireUtils-Schnittstelle....232
IMHReportEngine-Schnittstelle....234

IMHReportFormat-Schnittstelle....237
IMHReportParameter-Schnittstelle....238
IMHReportParameterList-Schnittstelle....239
IMHReportTemplate-Schnittstelle....239
IMHSelection-Schnittstelle....240
IMHSQLDef-Schnittstelle....241
IMHSystemUser-Schnittstelle....242
IMHUnmergedMessage-Schnittstelle....243
IMHWebPublicFile-Schnittstelle....247

Kapitel 11: Dialogue Server-API....249
Dialogue Server-API....250
Activity-API....251
Dialogue-API....255
Customer-API....263
Generic-API....271
Message-API....274
Emarketing Mail-API....287
Quest-API....289
Report-API....293
Selection-API....298
System-API....301
Telemarketing-API....305
Web Utilities-API....308

Kapitel 12: HQ Administration....313
Portrait Shared Server....314
Konfigurieren von Portrait HQ....314
Konfigurieren der Bildwiederholrate....314
Konfigurieren der Portrait HQ-Anwendungsprotokollierung....314
Konfigurieren der Portrait Shared Repository-Datenbank....315
Konfigurieren von SharePoint....315
Konfigurieren von Portrait Shared Server....315
Konfigurieren der Windows-Authentifizierung 315
Konfigurieren der Portrait Shared Server-Protokollierung....316
Konfigurieren der Kampagnen-Berichte....317
Konfigurieren der Quicklinks in MyView....318
Aktivieren von SSL/HTTPS....319

Aktivieren der Kampagnengenehmigung.....321
Ändern der HQ-Benutzerberechtigungen.....321
Marketingaktivitäten konfigurieren.....323
Aktivitätstypen konfigurieren.....323
Konfigurieren von Aktivitätsuntertypen.....324
Konfigurieren von Aktivitäteneigenschaften.....324
Konfigurieren von Kanälen.....325
Konfigurieren von Kundenkarten.....327
Ergebnisdaten-Integration.....330
Operationale Relationen.....330
Zusammenfassungsrelationen und Berichtsschema.....336
Protokollierung von Daten aus externen Tools.....342
HQ-Fehlerbehebung.....342
Fehler beim Authentifizieren bei Portrait HQ nach der Installation von SharePoint 342
Probleme mit den Sicherheitsanmeldeinformationen beim Ausführen von Portrait HQ.343
Fehlerbehebung bei einer „hängenden“ Anwendung beim Versuch der Anmeldung...343
Probleme beim Laden von Portrait Shared Services.....343
Erstellen von Dienstprinzipalnamen für Portrait Shared Services 343
Protokollierung.....344
Deinstallieren von Dialogue Server nach der Installation von SharePoint und Portrait HQ.344
Ändern das als Dienstkonto verwendete Konto nach der Installation von PSS 345
Erstellte Aufgaben werden nicht unter „Meine Ansicht“ in Portrait HQ angezeigt.....345
Anzeige der Fehlermeldung „Fehler beim Abrufen der Aufgabenliste“ in Portrait HQ..345

Kapitel 13: Hinweise zu Drittanbietern.....347
Hinweise zu Drittanbietern.....348

Kapitel 1

Dialogue Admin

In diesem Abschnitt:

- Dialogue Admin 12
- Dialogue Server-Hosts 12
- Datenbankinstanzen 12
- Kundendomänen 15
- Dialogeinrichtung 15
- Aktivitätseinrichtung 18
- Kanaltypen 19
- Ressourcen 22
- Benutzer und Sicherheit 25
- Allgemeine Verwaltung 32

Dialogue Admin

Dialogue Admin ist die administrative Client-Anwendung von Portrait Dialogue.

Dialogue Server-Hosts

Über das Dialogue Admin-Browserfenster können Sie sich mit einem oder mehreren Dialogue Server-Hosts verbinden. Ein *Dialogue Server-Host* steht für eine Installation auf einem Server.

Verwenden Sie für die Verbindung mit Dialogue Server den Knoten **Dialogue Server Hosts**, und wählen Sie aus dem Menü die Option **Element** > **Neu**.

Bearbeiten der Host-Eigenschaften

Die meisten Informationen unter **Host-Eigenschaften** sind schreibtgeschützte Informationen zu Softwareversion und Lizenz. Die folgenden Informationen können jedoch bearbeitet werden:

Englisch (USA) als Serversprache erzwingen (für alle Instanzen): Diese Option setzt die Sprache von Dialogue Server und all seinen Instanzen außer Kraft, so dass der Server Fehlermeldungen und anderen Anzeigetext in US-Englisch generiert. Diese Option ist für Systemadministratoren nützlich, die Dialogue Server zur kurzzeitigen Verwendung von US-Englisch für die Anzeige von Fehler- und Protokollinformationen einstellen möchten.

Hinweis: Dies entspricht der Einstellung unter **Datenbankinstanzeigenschaften**, wo die Sprache einer einzelnen Instanz übergangen werden kann.

Datenbankinstanzen

Innerhalb eines Dialogue Server-Hosts können mehrere Installationen von Dialogue Database enthalten sein. Jede dieser Installationer ist eine *Datenbankinstanz*.

Alle verfügbaren Instanzen werden unter dem Knoten **Datenbankinstanzen** aufgelistet.

Hinweis: Der Benutzer muss sich an einer Instanz mit einem gültigen Benutzernamen und Kennwort anmelden, um auf sie zugreifen zu können oder sie zu verwalten.

Definieren einer neuen Instanz

So definieren und richten Sie eine neue Instanz ein:

- Stellen Sie sicher, dass das Dialogue Server-Datenbankschema installiert ist.

- Wählen Sie den Knoten **Datenbankinstanzen**, und wählen Sie im Menü die Optionen **Element** > **Neu**. Dieser Vorgang kann nur lokal auf dem Server durchgeführt werden, auf dem Dialogue Server installiert ist.
- Melden Sie sich mit dem standardmäßigen Benutzernamen und dem dazugehörigen Kennwort bei der Instanz an (siehe Installationshandbuch).

Bei der Definition oder Bearbeitung einer Instanz können mehrere wichtige Eigenschaften konfiguriert werden:

- **Kurzkennung (ISID)**

  Eine Kurzkennung für Ihre Instanz. Die Kurzkennung muss aus ein oder zwei Großbuchstaben bestehen und sich von allen anderen Instanzen auf dem Dialogue Server-Host unterscheiden.

- **Ist die Standardinstanz**

  Standard ist aktiviert, wenn die ausgewählte Instanz die System-Standardinstanz ist (im Dialogue Server-Host).

- **Service Dialog Manager aktivieren**

  Aktiviert oder deaktiviert den Service Dialog Manager. Dieser Dienst ist u. a. für die Ausführung von Hintergrundaufgaben wie geplante und von Ereignissen ausgelöste Operationen verantwortlich.

- **Nachrichtenversanddienst aktivieren**

  Aktiviert oder deaktiviert den Nachrichtenversanddienst. Dieser Dienst ist verantwortlich für das Versenden der vom System ausgehenden Nachrichten.

- **Nachrichtenempfangsdienst aktivieren**

  Aktiviert oder deaktiviert den Nachrichtenempfangsdienst. Dieser Dienst ist für den Abruf und die Handhabung von unzustellbaren E-Mails verantwortlich.

- **Windows-Authentifizierung aktivieren**

  Die Aktivierung dieser Option erlaubt es Benutzern, sich mit ihren Windows-Anmeldeinformationen anzumelden und somit nicht mehr Benutzernamen und Kennwort angeben zu müssen.

- **Serverfehler protokollieren und Nachrichten debuggen**

  Diese Option aktiviert die Protokollierung aller Fehler und Ausnahmen in Dateien im Ordner:

  ```
  <DialogServer>\LogFiles\Errors
  ```

  Diese Option aktiviert außerdem die Protokollierung verschiedener Informationen im Portrait Dialogue Process Monitor. Am wichtigsten ist die detaillierte Protokollierung von SQL-Fehlernachrichten und die Aktivierung der Methode *LogDebugMessage* der Plug-In-API.

- **Englisch (USA) als Serversprache erzwingen**

  Diese Option setzt die Sprache der Instanz oder Benutzersitzung außer Kraft, so dass Dialogue Server Fehlermeldungen und anderen Anzeigetext in US-Englisch generiert. Diese Option ist für Systemadministratoren nützlich, die eine Instanz zur kurzzeitigen Verwendung von US-Englisch für die Anzeige von Fehler- und Protokollinformationen einstellen möchten.

  Hinweis: Es gibt eine entsprechende Einstellungen unter **Host-Eigenschaften**, mit der die Serversprache für alle Instanzen außer Kraft gesetzt werden kann.

- **Instanz offline schalten**

  Diese Option deaktiviert die Instanz für die Benutzung und serverseitige Verarbeitung. Wenn die Option aktiviert ist, können Benutzer sich nicht anmelden oder Servermethoden aufrufen. Bereits laufende Methodenaufrufe werden jedoch nicht unterbrochen. Diese Option ist beispielsweise nützlich, wenn Sie die Datenbank pflegen oder ein System-Upgrade durchführen.

- **Datenbankservertyp**

  Eine der unterstützten DBMS (derzeit Oracle und MS SQL Server).

- **Verbindungszeichenfolge**

  Zur Verbindung mit Dialogue Database wird die MS OLE DB-Verbindungzeichenfolge verwendet.

  Hinweis: Wenn Sie SQL Server Native Client zur Verbindung verwenden, müssen Sie den Parameter `Persist Security Info` auf der Registerkarte **Alle** im Dialog „Verbindungszeichenfolge“ auf `True` setzen, um das Benutzerkennwort für die Datenbank dauerhaft verwenden zu können.

- **Sprache der Systemdatenbank**

  Die Sprache, in der Dialogue Database und die Systemdaten installiert wurden. Dies ist auch die Standardsprache für die Benutzeroberfläche dieser Instanz.

  Hinweis: Diese Einstellung muss mit der tatsächlichen Sprache von Dialogue Database übereinstimmen. Klicken Sie auf die Schaltfläche **Beheben**, um die Spracheinstellung aus der Datenbanktabelle SYSTEM_INFO abzurufen. Wenn Sie die Verbindungszeichenfolge angeben oder ändern, versucht Dialogue Admin automatisch eine Verbindung mit der Datenbank aufzubauen und die Sprache abzurufen.

- **Wiederholter Verbindungsversuch**

  Der **Minimalwert** bestimmt die Zeitdauer vor einem erneuten Verbindungsversuch für den Fall, dass Dialogue Database nicht verfügbar ist. Dieses Intervall verdoppelt sich mit jedem weiteren Verbindungsversuch bis der **Maximalwert** erreicht ist, der danach als Intervallwert für Neuversuche verwendet wird.

Bearbeiten einer Instanz

Wählen Sie zum Bearbeiten der Eigenschaften einer Instanz die Instanz im Dialogue Admin-Browser aus, und wählen Sie im Menü die Optionen **Element** > **Eigenschaften**.

Neustarten einer Instanz

Wählen Sie zum Neustarten einer Instanz die Instanz im Dialogue Admin-Browser aus, und wählen im Menü die Optionen **Element** > **Neustart**.

Diese Neustartoption wird v. a. verwendet, wenn eine Instanz ausfällt, beispielsweise wenn die Datenbank nicht mehr verfügbar ist. Eine Instanz neu zu starten signalisiert den Dialogue Server-Diensten (MH Dialog Manager-Dienst und MH-Nachrichtenversanddienst), dass die Instanz wieder benutzt werden kann.

Der Neustart einer Instanz löscht außerdem alle zwischengespeicherten Daten in dieser Instanz, inklusive SQL-Abfragen, Kundendomänendefinitionen, Parameter usw.

Kundendomänen

Kundendomänen enthalten alle Kundensätze, die in Portrait Dialogue definiert wurden. Ein solcher Kundensatz wird als „*Kundendomäne*“ bezeichnet. Beispiele für Kundendomänen sind: Personen, B2B-Markt, Haushalte oder potentielle Kunden.

Weitere Informationen und Hilfe zu Kundendomänen finden Sie im Abschnitt *Kundendomänen und Ausdrücke* in der **Technischen Hilfe**.

Kategorien

Kategorien werden zur Beschreibung von Kunden verwendet. Ein Kunde kann Mitglied einer oder mehrerer Kategorien sein. Es gibt die folgenden vier Kategorietypen:

- **Einfache Kategorie**

  Ein Kunde kann Mitglied einer einfachen Kategorie sein oder auch nicht.

- **Kategorie mit Werten**

  Wenn ein Kunde Mitglied einer Kategorie mit Werten ist, bekommt er einen oder mehrere Werte in dieser Kategorie zugewiesen. Wenn in einer solchen Kategorie die Mehrfachauswahl zulässig ist, können einem Kunden mehrere Werte zugewiesen werden. Anderenfalls wird dem Kunden nur genau ein Wert zugewiesen.

- **Bewertungskategorie**

  Wenn der Kunde Mitglied einer Bewertungskategorie ist, besitzt der Kunde einen Bewertungswert zwischen einem definierten Minimal- und Maximalwert.

- **Kanal-Blockierungskategorie**

  Wenn ein Kunde Mitglied einer Kanal-Blockierungskategorie ist, wird er weder Nachrichten durch den angegebenen Kanal empfangen, noch findet sonstige Kommunikation durch diesen Kanal statt. Beispiel: Wenn ein Kunde Mitglied einer E-Mail-Blockierungskategorie ist, werden keine E-Mails an diesen Kunden gesendet.

Dialogeinrichtung

„Dialogeinrichtung“ enthält Unterpunkte zur Administration von Elementen, die für die Dialogerstellung verwendet werden.

Gruppentypen

Gruppentypen sind definierte Typen von Gruppen, die in Dialogen verwendet werden können. Ein Gruppentyp hat einen Namen und einen Datentyp. Der Gruppendatentyp kann einer der folgenden drei Typen sein:

- **Datenbank**: Der Datentyp „Datenbank“ stellt alle Kunden in Ihrer Kundendatenbank dar, genauer: alle Kunden in der von diesem Dialog betroffenen Kundendomäne.
- **Standard**: Der Datentyp „Standard“ ist der Datentyp, in dem die Dialogteilnehmer im Normalfall gespeichert sind.
- **Fuzzy**: Der Datentyp „Fuzzy“ wird für nicht identifizierte Kunden, beispielsweise einen Markt oder eine Zielgruppe, verwendet.

Sie können eine beliebige Anzahl von Gruppentypen definieren, um Ihre Dialoge nach Ihren Wünschen zu gestalten. Die Funktionen einer Gruppe werden dabei immer für den von Ihnen gewählten Datentyp bestimmt.

Operationstypen

Operationstypen sind die Definitionen der Operationen, die bei der Dialogerstellung in Visual Dialogue verwendet werden.

Es kann eine beliebige Anzahl von Operationstypen definiert werden. Die Logik dieser Operationen wird in Plug-Ins implementiert. Es gibt standardmäßige Plug-Ins und programmierbare benutzerdefinierte Plug-Ins. Die Verbindung zwischen Operation und Plug-In wird innerhalb eines Operationstyps eingestellt. Eine solche Verbindung wird als „Verzweigungstyp“ bezeichnet. Die Anordnung der Verzweigungstypen in einem Operationstyp kann konfiguriert werden.

Außerdem können das für eine Operation verwendete Symbol, die Operationskategorie (für die visuelle Kategorisierung in Visual Dialogue) sowie die standardmäßigen Ausführungsoptionen konfiguriert werden.

Dialogstatustypen

Dialogstatustypen dienen nur zur Kategorisierung von Dialogen gemäß den verschiedenen Stufen eines Dialog-Lebenszyklus. Ein Dialog besitzt immer einen der definierten Statustypen.

Es kann eine beliebige Anzahl von Statustypen definiert werden. Einige Typen sind jedoch Systemstandard und können daher nicht entfernt werden.

Ereignistypen

Ereignistypen sind Definitionen von Ereignissen, die Dialogue Server von außerhalb gemeldet werden.

Einige Ereignistypen sind standardmäßig vorhanden und werden vom System selbst verwendet. Andere Ereignistypen können in Integrationsprojekten hinzugefügt werden, um ein externes Ereignis in einer anderen Anwendung so zu definieren, dass Dialogue Server es registriert.

Ereignisse können durch die Dialogue Server-API oder direkt in Dialogue Database mit dem Aufruf einer gespeicherten Prozedur eingegeben werden.

Anrufstatustypen

Mit einem *Anrufstatustyp* wird das Resultat eines Anrufs beim Kunden im Rahmen eines Telemarketing-Projekts beschrieben. So kann verfolgt werden, ob ein Anruf erfolgreich oder kein Anschluss unter der Kundennummer vorhanden war.

Es kann eine beliebige Anzahl von Anrufstatustypen definiert werden. Einige Typen sind jedoch Systemstandard und können daher nicht entfernt werden.

Einrichtung für unzustellbare E-Mails

„Einrichtung für unzustellbare E-Mails“ enthält Unterpunkte zur Einrichtung der Handhabung unzustellbarer E-Mails.

Unzustellbarkeitskonfigurationen

Dialogue Server kann automatisch unzustellbare E-Mails handhaben. Sie können verschiedene Unzustellbarkeitskonfigurationen einrichten, nach denen der *MH-Nachrichtenempfangsdienst* E-Mails aus diversen Konten empfängt und verarbeitet.

Wenn Sie eine Konfiguration erstellen oder bearbeiten, werden Ihnen in einem Dialogfeld fünf Seiten angezeigt.

Allgemein: Hier können Sie den Namen der Konfiguration festlegen und diese aktivieren oder deaktivieren. Die Konfiguration muss aktiviert werden, damit eingehende E-Mails verarbeitet werden. Sie müssen außerdem Feldzuordnungen für die Kundendomäne einrichten. Diese Zuordnungen teilen dem Dienst mit, welche Felder welcher Domänen benutzt werden sollen, um den Empfänger der unzustellbaren E-Mail zu ermitteln.

Unzustellbarkeitslogik: Wenn Sie *Nach Kundennachricht-ID scannen* aktivieren, sucht der Nachrichtenempfangsdienst in der unzustellbaren E-Mail nach der Kundennachricht-ID der ursprünglichen E-Mail. *Scanzeitrahmen in Tagen* legt die Anzahl der Tage fest, während der der E-Mail-Verlauf überprüft wird, um eine nicht zugestellte E-Mail zu identifizieren. *Beim Scannen zu ignorierende Adressen* ist eine durch Semikolons getrennte Liste von Adressen, die beim Scannen von unzustellbaren E-Mails ignoriert werden sollen. Üblicherweise werden hier die Absenderadressen angegeben, die zum Versand der E-Mails verwendet werden.

Im Bereich *Nicht automatische Handhabung* können Sie einstellen, wie unzustellbare E-Mails behandelt werden, wenn manuelle Eingriffe nötig sind. *An diese Adresse weiterleiten* stellt die E-Mail-Adresse dar, an die unzustellbare E-Mails zur manuellen Behandlung weitergeleitet werden. Wenn Sie *Wenn möglich beim Weiterleiten Originalabsender verwenden* aktivieren, wird bei der Weiterleitung die Adresse des Originalabsenders verwendet. Dies ist besonders nützlich, wenn unzustellbare E-Mails nicht wirklich unzustellbar waren, sondern Antworten des Empfängers sind. Wenn dieses Kontrollkästchen nicht aktiviert ist oder die Originaladresse nicht gefunden wurde, wird die *Standardmäßige Absenderadresse* verwendet.

Unzustellbarkeitscodes: Hier können Sie die Einstellungen unter *Unzustellbarkeitscodes* überschreiben.

Mail-Server: Hier können Sie einstellen, wie sich der Dienst mit Mail-Servern verbindet.Der *Mail-Eingangsserver* wird zum Empfang unzustellbarer E-Mails verwendet. Das hier eingerichtete Konto ist üblicherweise dasselbe, das auch zum E-Mail-Versand benutzt wird. Sie können zwischen drei Eingangsservertypen wählen:

- **POP** unterstützt das Protokoll POP3.
- **IMAP** unterstützt das Protokoll IMAP4.
- Mit „**Keine**“ empfangen Sie überhaupt keine unzustellbaren E-Mails und müssen hierfür ein externes Programm verwenden. So können Sie eine eigene Lösung entwickeln und hängen hierfür nicht vom *MH-Nachrichtenempfangsdienst* ab. Die E-Mails müssen im MIME-Standard formatiert sein und die standardmäßigen SMTP-Zeichen für das Ende der E-Mail (CRLF.CRLF) enthalten. Jede E-Mail muss

in einer separaten Datei mit eindeutigem Namen in dieser Unzustellbarkeitskonfiguration gespeichert werden. Die Dateien müssen unter „<*Arbeitsverzeichnis*>\<*Konfigurationsname*>\Incoming“ gespeichert werden, wobei „*Arbeitsverzeichnis*“ das Verzeichnis ist, das in den allgemeinen E-Mail-Unzustellbarkeitssystemparametern als *Arbeitsverzeichnis* angegeben ist. *Konfigurationsname* ist der Name der Unzustellbarkeitskonfiguration.

Mail-Ausgangsserver wird zur Weiterleitung von E-Mails und dem Versand von Berichten verwendet.

Andere: Wenn *Kundeninfo immer protokollieren* aktiviert ist, wird der Nachrichtenempfangsdienst die unzustellbare E-Mail mit dem entsprechenden Kunden verknüpfen und diese Information in Dialogue Database protokollieren. Wenn diese Option deaktiviert ist, wird der Nachrichtenempfangsdienst nur die Information über die unzustellbare E-Mail ohne jede Kundeninformation protokollieren. Für Unzustellbarkeitscodes, deren Unzustellbarkeitsaktion auf „Ereignis“ gestellt ist, werden alle unzustellbaren E-Mails mit Kundeninformationen protokolliert, unabhängig von dieser Einstellung. Das Aktivieren dieser Option kann die Leistung Ihres Systems beeinflussen.

Adresse des Berichtsempfängers ist die E-Mail-Adresse des Empfängers, der den Bericht des Nachrichtenempfangsdienstes empfängt. Dieser Bericht enthält die Gesamtanzahl der unzustellbaren E-Mails und den jeweiligen Unzustellbarkeitscode. Sie können das Intervall des Berichtversandes mit *Intervall in Minuten* konfigurieren.

Der Nachrichtenempfangsdienst speichert die abgerufenen Nachrichten auf der Festplatte. Dies kann viel Speicherplatz benötigen, wenn Ihr System eine große Anzahl unzustellbarer Nachrichten erhält. Um Platz zu sparen, können Sie einen Dienst einrichten, der alte Dateien löscht. Verwenden Sie dazu „*Intervall in Tagen*“.

Unzustellbarkeitscodes

Wenn eine E-Mail unzustellbar ist, kann das verschiedene Gründe haben. Dialogue Server definiert daher verschiedene *Unzustellbarkeitscodes*. Jeder Unzustellbarkeitscode steht für einen Grund, der eine E-Mail unzustellbar macht. E-Mails mit unterschiedlichen Codes können vom System unterschiedlich behandelt werden.

Die *Unzustellbarkeitsaktion* legt fest, wie mit E-Mails mit einem bestimmten Unzustellbarkeitscode verfahren wird. Es gibt drei mögliche Aktionstypen:

- *Ignorieren* – Die unzustellbare E-Mail wird ignoriert und das System unternimmt nichts.
- *Ereignis* – Wenn das System den Originalempfänger der E-Mail identifizieren kann, wird ein Ereignis gemeldet.
- *Weiterleiten* – Die unzustellbare E-Mail wird an eine bestimmte Adresse weitergeleitet.

Nachrichtenprotokoll aktualisieren bestimmt, ob die Datenbanktabelle MESSAGE_LOG aktualisiert wird, wenn eine E-Mail als unzustellbar zurückkommt. Diese Aktualisierung wird nur durchgeführt, wenn das System die Ursprungsnachricht aus der zurückgekommenen, unzustellbaren Nachricht identifizieren kann.

Aktivitätseinrichtung

„Aktivitätseinrichtung“ enthält Unterpunkte zur Konfiguration von Aktivitätstypen und Aufgabenverhalten.

Aktivitätstypen

Aktivitätstypen sind Definitionen von Kundenaktivitäten, die in Dialogue Database gespeichert werden. Eine Aktivität bezeichnet die Interaktion mit einem Kunden.

Es kann eine beliebige Anzahl von Aktivitätstypen definiert werden. Sie können bei der Einrichtung eines Aktivitätstypen bestimmen, ob ein Benutzer diese Aktivität manuell über die Webanwendung einfügen kann. Einige Aktivitäten werden automatisch gemeldet, wenn Dialoge ausgeführt werden, während andere mittels Webanwendungen, der Dialogue Server-API oder anderen Anwendungssystemen gemeldet werden. Es steht eine gespeicherte Prozedur zur Verfügung, die Aktivitäten direkt in Dialogue Database registrieren kann.

Optionen für Aufgabennachverfolgung

Optionen für Aufgabennachverfolgung sind Standardwerte für die Einstellung der Nachverfolgungsdauer einer Aufgabe. Jeder Optionensatz kann definiert und über die Benutzeroberfläche und eine Dropdownliste verfügbar gemacht werden.

Aufgabenanzeigeintervalle

Aufgabenanzeigeintervalle sind Standardwerte für das Filtern der Liste *Meine Aufgaben* in Customer View. Es können beliebige Intervallsätze definiert und auf der Benutzeroberfläche aufgelistet werden.

Aufgabenfelder

Aufgabenfelder sind eine Reihe von Feldern, die Datenfeldern in Kundendomänen zugeordnet sind.

Eine Aufgabe ist einem Kunden in einer Kundendomäne zugeordnet (oder unter ihm registriert). Wenn eine Liste von Aufgaben angezeigt wird, können diese Aufgaben Kunden unterschiedlicher Domänen zugeordnet sein. Um Kundeninformationen in einer solchen Liste anzuzeigen, wurde das Konzept von *Aufgabenfeldern* eingeführt. Ein Aufgabenfeld wird Datenfeldern in Kundendomänen mit Aufgaben zugeordnet. Wenn das System Aufgabenlisten mit Aufgabenfeld-Spalten generiert, werden Daten aus verschiedenen Domänen zusammengetragen und in diesen Spalten angezeigt.

In Dialogue Database definiert die Tabelle TASK_FIELD die Aufgabenfelder, während in `TASK_FIELD_MAPPING` die Referenzen zu den Domänenfelder gespeichert sind.

Aufgabenfelder werden sowohl im Task Manager in Visual Dialogue als auch in „Meine Aufgaben“ in Customer View verwendet.

Kanaltypen

Kanaltypen sind alle Medien sowie andere Wege und Möglichkeiten, mit dem Kunden zu kommunizieren.

Einige Kanäle sind physische Kommunikationskanäle wie E-Mail, Fax oder SMS. Andere Kanäle stellen *Kommunikationssituationen* dar, etwa persönliche Treffen.

Bei der Installation von Dialogue Database werden eine Reihe von standardmäßigen Kanaltypen installiert. Sie können aber auch nach Ihren Wünschen konfiguriert werden.

Ausgabekanäle

Ein Ausgabekanal ist eine physische Implementierung eines Kanals für ausgehende Kommunikation.

So implementiert der zum E-Mail-Kanaltyp gehörende Ausgabekanal zum Beispiel die Logik zum Versand einer E-Mail über das SMTP-Protokoll.

Für einige Kanaltypen können mehrere physische Implementierungen passend sein. So können für das Versenden von SMS abhängig vom SMS-Dienstanbieter (der für den Versand von SMS zuständigen Telefongesellschaft) mehrere Implementierungen passen.

Ein Plug-In für einen Ausgabekanal definiert einen Satz von Steuerparametern. Einige dieser Parameter sind festgelegt und in den Eigenschaften des Ausgabekanals konfiguriert. Andere Steuerungsparameter hingegen werden für jede Nachrichtenvorlage konfiguriert. Die folgenden Absätze beschreiben die festgelegten Steuerparameter, die von den installierten Kanälen verwendet werden.

Feste Steuerparameter für E-Mail-Nachrichten

Die folgenden festen Steuerparameter werden im E-Mail-Ausgabekanal verwendet. Die meisten Werte für Steuerungsparameter sind durch die Standard-E-Mail-Protokolle festgelegt. Allerdings gibt es auch einige spezifische Parameter für Dialogue Server.

SMTP-Host-IP	Der Name oder die IP-Adresse des SMTP-Servers, von dem die E-Mails versendet werden. Dieser Parameter ist erforderlich.
SMTP-Hostport	Die Portnummer, die vom SMTP-Server verwendet wird. Standard ist 25. Dieser Parameter ist erforderlich.
Verbindungstimeout	Der Zeitüberschreitungswert in Millisekunden wird bei den Verbindungsversuchen zum SMTP-Server verwendet. Dieser Parameter ist erforderlich.
Benutzername	Der Benutzername wird bei der Anmeldung am SMTP-Server verwendet, wenn eine Authentifizierung erforderlich ist.
Passwort	Das Kennwort wird bei der Anmeldung am SMTP-Server verwendet, wenn eine Authentifizierung erforderlich ist.
Mail-Agent	Dieser optionale Parameter gibt das Programm an, das zum Erstellen von E-Mail-Nachrichten verwendet werden soll. Standard ist Dialogue Server.
Standardzeichensatz	Gibt den Standardzeichensatz an, der für E-Mail-Nachrichten verwendet wird. Standardzeichensatz ist UTF-8
Codierung für die Inhaltsübertragung	Gibt die für den Versand der E-Mail standardmäßig genutzte Codierung an. Mögliche Werte sind: Quoted-Printable, Base64 und 7bit. Der Standard ist Quoted-Printable.
QuotedPrintable RFC2045	Gibt an, ob die Nachricht RFC 2045-kompatibel sein soll. Der Standardwert lautet True.

Bounce-E-Mail-Adresse	Gibt den Wert für das Feld „Return-path“ im E-Mail-Header an. Standardmäßig (wenn dieses Feld leer ist) ist dieser Wert identisch mit der Absenderadresse, die im Feld „Von“ des E-Mail-Headers angegeben wird. Dieser Parameter dient üblicherweise dazu, alle unzustellbaren E-Mails in einem Konto zu sammeln.
Bounce für Fehler generieren	Gibt an, ob eine E-Mail mit der Unzustellbarkeitsnachricht generiert werden und an das zum Empfang unzustellbarer E-Mails konfigurierte E-Mail-Konto gesendet werden soll. Falls dies auf *True* festgelegt ist, werden E-Mails mit der Unzustellbarkeitsnachricht gemäß den Parametern *Hard-Bounce-Fehlercodes* und *Soft-Bounce-Fehlercodes* versendet.
Hard-Bounce-Fehlercodes	Eine durch Kommata getrennte Liste von SMTP-Fehlercodes, die als dauerhaft unzustellbar gehandhabt werden.
Soft-Bounce-Fehlercodes	Eine durch Kommata getrennte Liste von SMTP-Fehlercodes, die als temporär unzustellbar gehandhabt werden.
Ausgabeziel	Gibt das Ausgabeziel des Plug-Ins zum Versenden von E-Mails an. Mögliche Werte sind: „Keins“, „SMTP“ und „Datei“. Der Standardwert ist „SMTP“.
Ausgabepfad	Gibt den vollständigen Pfad zum Speicherort der E-Mail-Dateien an, wenn „Datei“ als Ausgabeziel eingestellt wurde.

Feste Steuerparameter für Facebook-Nachrichten

Für die Facebook-Ausgangskanäle gibt es keine festgelegten Steuerparameter. Alle Facebook-Steuerparameter werden für die einzelnen Nachrichtenvorlagen konfiguriert.

Feste Steuerparameter für FTP-Nachrichten

Für den FTP-Ausgangskanal gibt es keine festgelegten Parameter. Alle FTP-Steuerparameter werden für die einzelnen Nachrichtenvorlagen konfiguriert.

Feste Steuerparameter für SMS-Nachrichten

Die folgenden Steuerungsparameter werden für SMS-Textnachrichten verwendet.

Server-URL	Eine URL zum SMS-Gateway, der von einem Dienstanbieter zum SMS-Versand gestellt wird. Die Server-URL ist erforderlich.

Hinweis: Die Steuerparameter für SMS-Nachrichtentypen können von Installation zu Installation abweichen, da das System für unterschiedliche SMS-Versandanbieter eingerichtet ist. Die hier aufgeführten Parameter werden vom Anbieter des ScanGIT SMS Gateway in Norwegen verwendet.

Feste Steuerparameter für Twitter-Nachrichten

Für die Twitter-Ausgangskanäle gibt es keine festgelegten Steuerparameter. Alle Twitter-Steuerparameter werden für die einzelnen Nachrichtenvorlagen konfiguriert.

Nachrichtentypen

Nachrichtentypen definieren, wie Nachrichten erstellt werden. Eine Vorlage oder eine Basisnachricht basiert immer auf einem Nachrichtentyp.

Ein Nachrichtentyp verwendet ein Plug-In, um die Nachricht zu erstellen. Normalerweise führt das Plug-In die Nachrichtenvorlage oder die Basisnachricht mit den Kundendaten (aus einer Kundendomäne) zusammen.

Ein Nachrichten-Plug-In kann angepasst werden. Mit Dialogue Server wurde bereits ein Satz von Standard-Plug-Ins installiert:

- Text-Seriendruck-Plug-In
- RTF-Seriendruck-Plug-In
- XSL-Transformations-Plug-In
- Microsoft Word-Seriendruck-Plug-In

Beim Einrichten von Nachrichtentypen kann eine gewisse Anzahl von Eigenschaften konfiguriert werden. Die Wichtigsten sind:

- Ein optionaler Ausgangskanal, der zum Versand von Nachrichten diesen Typs verwendet wird.
- Das Plug-In zum Erstellen (Zusammenstellen) der Nachricht.
- Der standardmäßige Aktivitätstyp, der generiert wird, wenn eine Nachricht dieses Typs erstellt wird.
- Ob der integrierte Nachrichteneditor oder ein externer Editor zur Bearbeitung von Vorlagen und Basisnachrichten verwendet werden soll.

Optimierung der Nachrichtenspeicherung

Die Option *Speicheroptimierung zulassen* legt fest, ob die Art der Speicherung der Nachrichten in der Datenbank optimiert werden kann, um so die Menge der gespeicherten Daten zu verringern. Die Speicheroptimierung muss dafür vom verwendeten Nachrichten-Plug-In unterstützt werden. Das *Seriendruck-Plug-In* und das *RTF-Seriendruck-Plug-In* unterstützen die Speicheroptimierung. Falls *Speicheroptimierung zulassen* deaktiviert ist, wird der endgültige Nachrichteninhalt immer gespeichert.

Ressourcen

Ressourcen enthält eine Sammlung von systemspezifischen Low-Level-Elementen. Diese Elemente werden üblicherweise von den Benutzern verwaltet, die sowohl für die Integration mit anderen Systemen und Datenquellen als auch für die Implementierung benutzerdefinierter Plug-Ins verantwortlich sind.

SQL-Repository

SQL-Repository enthält Gruppen benutzerdefinierter SQL-Befehle, die von Dialogue Server verwendet werden. Es gibt verschiedene Bereiche, in denen SQL-Anfragen hinzugefügt und angepasst werden können:

- Zum Datenempfang in *Kundendomänen*

- Zur Aktualisierung von Daten aus *Kundendomänen*
- Zur Ausführung von SQL-Befehlen aus *benutzerdefinierten Plug-Ins*
- Zur Implementierung von *Auswahl-* und *Divisionsoperationen* in Dialogen, die auf SELECT-Anweisungen basieren.

Hinweis: Um zusätzlich zu den benutzerdefinierten SQL-Abfragen die intern in Dialogue Server verwendeten SQL-Abfragen anzuzeigen, aktivieren Sie das Menüelement unter **Ansicht** > **Systemdaten**. Diese intern verwendeten SQL-Abfragen dürfen nicht geändert werden.

Sekundäre Datenbanken

Normalerweise werden SQL-Anweisungen über die *Standardverbindung* ausgeführt. Die Standardverbindung ist die OLE DB-Verbindung, die zur Verbindung mit Dialogue Database verwendet wird. Allerdings könnte Dialogue Server zur Abfrage anderer Daten in anderen Datenbanken auch andere Verbindungen benötigen. So kann eine Gruppe von Datensätzen einer Paradox-Datenbank beispielsweise eine Kundendatenbank darstellen. Solch eine alternative Datenbank wird als *sekundäre Datenbank* bezeichnet und hat eine eigene Datenbankverbindung.

Bei der Einrichtung einer sekundären Datenbank wird eine OLE DB-Verbindungszeichenfolge konfiguriert. Außerdem muss der *Datenbanktyp* angegeben werden, so dass Dialogue Server weiß, wie er SQL-Anweisungen formatieren muss.

Hinweis: Viele OLE DB-Treiber und durch OLD DB verfügbare ODBC-Treiber unterstützen keine MTS-Transaktionen und Connection Pooling. Dies kann zu Fehlern wie „*Failed to enlist in the caller's transaction*“ (Fehler bei Anwendung der Anrufertransaktion) führen. Um die automatische Transaktionsregistrierung und Connection Pooling zu überbrücken, fügen Sie das Folgende zur Verbindungszeichenfolge hinzu:

```
;OLE DB Services=-3
```

In einigen Fällen können die passenden Werte 0 oder -4 anstatt -3 sein. Beachten Sie, dass das Hinzufügen dieser Einstellung Transaktionen für *diese* Datenbank *deaktiviert*.

Webdienste

Webdienste enthält eine Reihe von Webdiensten, die dem Dialogue Server bekannt sind. Wenn der Server weiß, wie er einen Webdienst aufrufen kann, kann dieser Dienst als Datenquelle in einer Kundendomäne benutzt oder von einer Operation in einem Dialog aufgerufen werden.

Hinweis: Webdienste und ihre Anwendung sind in der aktuellen Version nicht vollständig implementiert.

Plug-In-Repository

Das *Plug-In-Repository* enthält Codeteile oder Plug-Ins, die benutzerdefinierte Funktionen implementieren. Ein Plug-In kann als *COM-Objekt*, *.NET-Klasse* oder *Skript* implementiert werden. Der Skriptansatz verwendet **Windows Script** und unterstützt installierte Skriptsprachen wie VBScript, JScript und PerlScript. Dialogue Server kann die Plug-Ins bei Bedarf laden und aufrufen. So werden Plug-Ins beispielsweise aufgerufen, wenn eine Dialogoperation ausgeführt wird.

Verschiedene Arten von Plug-Ins

Es gibt sechs Gruppen von Plug-Ins:

- **Kundendaten-Plug-Ins**

  Diese Plug-Ins werden in der Definition von Kundendomänen verwendet, um Kundendaten zu aktualisieren.

- **Plug-Ins für Dialogverzweigungen**

  Diese Plug-Ins implementieren Verzweigungen in Dialogoperationen**.** Viele der Standardverzweigungs-Plug-Ins, die mit dem System installiert werden, sind implementiert als Open-Source-Skripte.

- **Ausdrucks-Plug-Ins**

  Diese Plug-Ins implementieren benutzerdefinierte Ausdrucksfunktionen.

- **Nachrichten-Plug-Ins**

  Diese Plug-Ins sind zuständig für das Erstellen von Nachrichten. Eines der Standard-Plug-Ins fügt beispielweise Word-Vorlagen mit Kundendaten zusammen.

- **Ausgangskanal-Plug-Ins**

  Diese Plug-Ins implementieren die physische, ausgehende Kommunikation über einen Kanal. Beispielsweise sendet das Plug-In „E-Mail senden“ E-Mail-Nachrichten mithilfe des SMTP-Protokolls.

- **Generische Plug-Ins**

  Diese Plug-Ins implementieren benutzerdefinierte Logiken und können über die Dialogue Server-API mit einer beliebigen Anzahl von Eingabeparametern und einem Rückgabewert aufgerufen werden.

Diese vier Typen von Plug-Ins folgen unterschiedlichen Regeln für ihre Implementierung. Die von verschiedenen Plug-In-Typen implementierten Funktionen unterscheiden sich. Eine detaillierte Beschreibung finden Sie im Abschnitt *Plug-In-APIs*.

Installieren von Plug-Ins, die als COM-Objekte implementiert sind

Plug-Ins, die als COM-Objekte implementiert sind, müssen lokal auf dem Computer registriert werden, auf dem Dialogue Server ausgeführt wird. Normalerweise geschieht dies mit dem Windows-Programm **regsvr32.exe**. In Dialogue Admin muss die GUID, welche die Klassen-ID des COM-Objekts bezeichnet, im Fenster „Eigenschaften des Plug-Ins“ angegeben werden.

Installieren von Plug-Ins, die als .NET-Klassen implementiert sind

.NET-basierte Plug-Ins werden als Klassen in einer .NET-Klassenbibliothek erstellt. Der Dateiname und der Pfad der kompilierten Bibliothek müssen in Dialogue Admin im Fenster ***Eigenschaften des Plug-Ins*** angegeben werden. Dieser Pfad muss lokal auf den Computer verweisen, auf dem Dialogue Server ausgeführt wird. Es darf *kein* Netzwerkpfad sein. Außerdem muss der Name und der Namensraum der Klasse angegeben werden.

Für .NET-Plug-Ins kann eine Konfigurationsdatei (ähnlich wie app.config) angegeben werden. Die Konfigurationsdatei muss im selben Ordner wie das .NET-Assembly liegen und den selben Namen plus **.config** haben. Beispiel:Der Name der Konfigurationsdatei zum Assembly **myplugin.dll** muss **myplugin.dll.config** sein.

Für .NET-Plug-Ins steht die Option „Immer in einer separaten Anwendungsdomäne ausführen“ zur Verfügung. Wenn diese Option aktiviert ist, wird jede Instanz der Plug-In-Klasse in einer neuen .NET-

Anwendungsdomäne erstellt. Das Erstellen neuer Anwendungsdomänen erfordert umfangreichere Initialisierungen und kann die Systemleistung beeinträchtigen. Dies ist zum Beispiel der Fall, wenn WCF (Windows Communication Foundation) initialisiert werden muss. Diese Option ist jedoch nützlich, wenn die Instanzen des Plug-Ins voneinander getrennt werden müssen, beispielsweise aus Gründen der Thread-Sicherheit. Bitte beachten Sie, dass für das „MH-Plug-In zum Versenden von E-Mails“ diese Option aktiviert sein muss.

Benutzer und Sicherheit

Benutzer und Sicherheit enthält Unterpunkte zur Konfiguration der Systembenutzer, Benutzergruppen und deren Zugriffsrechten sowie der Objektsicherheit.

Benutzer

Benutzer sind die Betreiber (oder Personen) mit Zugang zu Portrait Dialogue. Ein Benutzer hat einen Benutzernamen, vollen Namen und ein Kennwort. Zusätzlich können optionale Eigenschaften wie E-Mail-Adresse und Mobiltelefonnummer eingerichtet werden.

Benutzer haben unterschiedliche Zugriffsrechte auf das System und können Mitglieder einer oder mehrerer Benutzergruppen sein. Ein Benutzer erbt die Zugriffsrechte der Gruppen, deren Mitglied er ist.

Der Benutzer „Intern“ wird intern verwendet und sollte nicht verändert werden.

Hinweis: Um einen Benutzer zu kopieren, wählen Sie den Benutzer aus, und wählen Sie **Element** > **Kopie erstellen** aus dem Menü.

Benutzergruppen

Eine *Benutzergruppe* ist eine Gruppe von Systembenutzern. Zugriffsrechte einer Gruppe werden an ihre Mitglieder weiter vererbt.

Zugriffsrechte

Die Tabelle unten beschreibt die unterschiedlichen Zugriffsrechts, die an Benutzer und Benutzergruppen vergeben werden können.

Modul	Zugriffsrecht	Beschreibung
Dialogue Server	Erstellen von Benutzersitzungen für andere Benutzer durch den Benutzer zulassen	Benutzer können sich mit der **LoginDelegate** -Methodeder API als ein anderer Benutzer anmelden.
Dialogue Admin	Anmeldung des Benutzers bei Dialogue Admin zulassen	Benutzer können sich an einer Instanz von Dialogue Admin anmelden.
	Änderung der Daten in Dialogue Admin durch den Benutzer zulassen	Benutzer können Änderungen in Dialogue Admin vornehmen, mit Ausnahme von Ände-

		rungen, die Auswirkungen auf Systembenutzer und Sicherheit haben.
	Änderung der Systemsicherheit durch den Benutzer zulassen	Benutzer können Veränderungen an der Systemsicherheit in Dialogue Admin vornehmen.
Dialogue	Anmeldung des Benutzers bei Dialogue zulassen	Der Benutzer kann sich bei Visual Dialogue anmelden.
	Erstellung neuer Dialoge durch den Benutzer zulassen	Benutzer können neue Dialoge in Dialogue erstellen.
	Bearbeitung von Dialogen durch den Benutzer zulassen	Benutzer können Dialoge in Dialogue bearbeiten und verändern.
	Löschen von Dialogen durch den Benutzer zulassen	Benutzer können Dialoge in Dialogue löschen.
	Ausführung von Dialogen durch den Benutzer zulassen	Benutzer können Dialogoperationen in der *Ausführungsansicht* eines Dialogs ausführen.
	Löschen von Dialogen durch den Benutzer zulassen	Benutzer können Dialoge von allen Ausführungsdaten bereinigen, was auch das Löschen aller Beteiligten eines Dialogs einschließt.
	Erstellung von Fragebögen durch den Benutzer zulassen	Benutzer können Fragebögen erstellen, verändern und löschen.
	Löschen von Antwortdaten durch den Benutzer zulassen	Benutzer können in Dialogue Antwortdaten löschen.
	Zugriff auf Aufgabenverwaltung durch den Benutzer zulassen	Benutzer können den Aufgabenorganisator in Dialogue öffnen und nutzen. Dies schließt Zugriffsrechts auf die Ansicht *Aufgabenübersicht* nicht mit ein.
	Zugriff des Benutzers auf die Aufgabenübersicht zulassen	Benutzer können auf die Ansicht „Aufgabenübersicht“ zugreifen (dies erfordert auch das Zugriffsrecht auf den Aufgabenorganisator).
	Erstellung neuer Berichtsvorlagen durch den Benutzer zulassen	Benutzer können in Dialogue neue Berichtsvorlagen erstellen.
	Bearbeitung von Berichtsvorlagen durch den Benutzer zulassen	Benutzer können in Dialogue Berichtsvorlagen bearbeiten und löschen.
	Löschen von E-Mail- und Verknüpfungsnachverfolgungsdaten durch den Benutzer zulassen	Benutzer können E-Mail- und Verknüpfungsnachverfolgungsdaten in Dialogue löschen.
	Erstellen von Mastervorlagen durch den Benutzer zulassen	Benutzer können Mastervorlagen erstellen, ändern und löschen.
	Erstellung von Inhaltsobjekten durch den Benutzer zulassen	Benutzer können Inhaltsobjekte erstellen, ändern und löschen.

	Export von Kundendaten durch den Benutzer zulassen	Benutzer können Kundendaten aus Dialogue exportieren.
	Export von Objekten durch den Benutzer zulassen	Benutzer können Objekte aus Dialogue exportieren.
	Import von Objekten durch den Benutzer zulassen	Benutzer können Objekte in Dialogue importieren.
Customer View	Anmeldung des Benutzers bei Customer View zulassen	Benutzer können sich am Webmodul „Customer View“ anmelden.
	Durchsuchen der Kundendaten durch den Benutzer zulassen	Benutzer können Kundendomänen nach Kunden durchsuchen und einzelne Kundendaten ansehen.
	Änderung der Kundendaten durch den Benutzer zulassen	Benutzer können neue Kunden hinzufügen und vorhandene Kunden bearbeiten.
	Löschen von Kunden durch den Benutzer zulassen	Benutzer können Kunden löschen.
	Bearbeitung von Aktivitäten durch den Benutzer zulassen	Benutzer können Aktivitäten eines Kunden hinzufügen und bearbeiten.
	Löschen von Aktivitäten durch den Benutzer zulassen	Benutzer können Aktivitäten eines Kunden löschen.
	Bearbeitung von Nachrichten durch den Benutzer zulassen	Benutzer können Nachrichten für einen Kunden hinzufügen und bearbeiten.
	Löschen von Nachrichten durch den Benutzer zulassen	Benutzer können die Nachrichten eines Kunden löschen.
	Ansicht von Antwortformularen durch den Benutzer zulassen	Benutzer können Antwortformulare ansehen, die auf einen Kunden registriert sind.
	Bearbeitung von Antwortformularen durch den Benutzer zulassen	Benutzer können Antwortformulare eines Kunden hinzufügen und bearbeiten.
	Löschen von Antwortformularen durch den Benutzer zulassen	Benutzer können Antwortformulare löschen, die auf einen Kunden registriert sind.
	Bearbeitung der Kategoriezugehörigkeit durch den Benutzer zulassen	Benutzer können Zugehörigkeiten zu Kategorien hinzufügen und bearbeiten.
	Löschen der Kategoriezugehörigkeit durch den Benutzer zulassen	Benutzer können die Zugehörigkeit zu einer Kategorie entfernen.
	Durchsuchen der Dialoge durch den Benutzer zulassen (Web)	Benutzer können die Dialoge durchsuchen, an denen ein Kunde teilnimmt.
	Bearbeitung der Dialogteilnahme durch den Benutzer zulassen	Benutzer können einen Kunden zu einem Dialog hinzufügen, wenn der Dialog Gruppen

		enthält, die das manuelle Einfügen von Beteiligten erlauben.
	Löschen der Dialogteilnahme durch den Benutzer zulassen	Benutzer können die Beteiligung eines Kunden an einem Dialog löschen.
Telemarketing Web	Zugriff des Benutzers auf Telemarketing-Projekte zulassen	Benutzer können sich am Webmodul „Telemarketing“ anmelden und auf jedes TM-Projekt zugreifen. Diese Benutzer können als Betreiber in allen Projekten handeln. **Hinweis:** Benutzer werden normalerweise durch den Telemarketing-Designer in Dialogue einzelnen Telemarketing-Projekten zugewiesen und haben nicht auf alle Projekte Zugriff.
	Zugriff auf TM-Projektberichte durch den Benutzer zulassen	Benutzer können die Telemarketing-Berichtsseite im Webmodul „Telemarketing“ anzeigen.
Report Portal	Anmeldung bei Report Portal durch den Benutzer zulassen	Benutzer können sich bei Report Portal anmelden.
	Ausführung von Berichten durch den Benutzer zulassen	Benutzer können Berichte aus Report Portal ausführen.
	Ansicht archivierter Berichte durch den Benutzer zulassen	Benutzer können archivierte Berichte im Report Portal ansehen.
	Speichern von Berichten im Archiv durch den Benutzer zulassen	Benutzer können von ihnen in Report Portal ausgeführte Berichts im Berichtsarchiv speichern.
	Löschen von Berichten aus dem Archiv durch den Benutzer zulassen	Benutzer können archivierte Berichts aus dem Berichtsarchiv in Report Portal löschen.
	Überschreiben von Systemparametern durch den Benutzer zulassen	Benutzer können die Parameter „Benutzer-ID“ und „Benutzername“ in Systemberichten ändern.
Dashboard	Anmeldung bei Dashboard für Benutzer zulassen	Benutzer können sich bei Dashboard anmelden.
	Änderung der eigenen Einstellungen des Dashboard durch den Benutzer zulassen	Benutzer können gemeinsame Widgets für sich konfigurieren sowie eigene Widgets hinzufügen und konfigurieren.
	Änderung der Standardeinstellungen des Dashboard durch den Benutzer zulassen	Benutzer können allen Nutzern gemeinsame Widgets hinzufügen und konfigurieren.

Web Portal	Zugriff auf den Nachrichten-Designer durch den Benutzer zulassen	Benutzer können sich beim Nachrichten-Designer anmelden.
	Bearbeitung von Nachrichten-Designer-Vorlagen durch den Benutzer zulassen	Benutzer können Vorlagen im Nachrichten-Designer erstellen und bearbeiten.
	Löschen von Nachrichten-Designer-Vorlagen durch den Benutzer zulassen	Benutzer können Vorlagen im Nachrichten-Designer löschen.
	Spam-Bewertung der Vorlagen durch den Benutzer zulassen	Benutzer können die Spam-Bewertungsfunktion im Nachrichten-Designer verwenden.
	Versenden von Testnachrichten durch den Benutzer zulassen	Benutzer können die Testversandfunktion im Nachrichten-Designer verwenden.
Outlook-Plug-In	Anmeldung am MS Outlook-Plug-In durch den Benutzer zulassen	Benutzer können sich am Outlook-Plug-In anmelden und das Plug-In nutzen.

Objektsicherheit

Objektsicherheit enthält Objekte verschiedener Typen, für die Sie sowohl die Eigentümerschaft als auch, falls Sperrung und Entsperrung anwendbar ist, deren Entsperrung veranlassen können.

Jedes Objekt verfügt über einen Eigentümer (*owner).* Der Eigentümer ist standardmäßig der Benutzer, der das Objekt erstellt hat.

Einige Objekttypen können von Benutzern gesperrt werden, die das Objekt momentan in Visual Dialogue bearbeiten. Wenn ein Objekt explizit entsperrt werden muss, kann dies im Fenster „Objekteigenschaften“ vom Systemadministrator verwaltet werden.

Außerdem können für einige Objekttypen die Zugriffsrechte individuell konfiguriert werden. Es gibt vier solcher Objekttypen:

- **Kundendomänen**

  Zugriffsrechte für Kundendomänen können detailliert eingerichtet werden. Die hier konfigurierten Zugriffsrechte gelten für Benutzer von Web Applications.

- **Dialoge**

  Zugriffsrechte für jeden Dialog können detailliert eingerichtet werden. Dies steuert die Zugriffsrechte innerhalb von Visual Dialogue.

- **Berichtsvorlagen**

  Zugriffsrechte für jede Berichtsvorlage können detailliert eingerichtet werden. Dies steuert sowohl den Zugriff im Berichtsvorlagen-Designer von Visual Dialogue als auch in der Webanwendung „Web Portal“.

- **Telemarketing-Projekte**

  Jedes Telemarketing-Projekt fungiert als ein Objekt. Es können Zugriffsrechte wie „Als Telemarketing-Betreiber agieren“ eingerichtet werden.

Beschreibung von Zugriffsrechten, die auf Objekte angewendet werden

Die folgende Tabelle beschreibt die verschiedenen Zugriffsrechte auf Objekte, die Benutzern und Benutzergruppen zugewiesen werden können.

Objekttyp	Zugriffsrecht	Beschreibung
Kundendomäne	Auf Kundendaten in Customer View zugreifen	Benutzer können in der Domäne nach Benutzern suchen und sich diese anzeigen lassen.
	Änderung der Kundendaten durch den Benutzer zulassen	Benutzer können in der Domäne neue Kunden hinzufügen und vorhandene Kunden bearbeiten.
	Löschen von Kunden durch den Benutzer zulassen	Benutzer können in der Domäne Kunden löschen.
	Bearbeitung von Aktivitäten durch den Benutzer zulassen	Benutzer können in der Domäne Aktivitäten von Kunden hinzufügen und bearbeiten.
	Löschen von Aktivitäten durch den Benutzer zulassen	Benutzer können in der Domäne Aktivitäten von Kunden löschen.
	Bearbeitung von Nachrichten durch den Benutzer zulassen	Benutzer können in der Domäne Nachrichten der Kunden hinzufügen und bearbeiten.
	Löschen von Nachrichten durch den Benutzer zulassen	Benutzer können in der Domäne Nachrichten der Kunden löschen.
	Zugriff auf Antwortformulare durch den Benutzer zulassen	Benutzer können in der Domäne Antwortformulare durchsuchen und anzeigen.
	Bearbeitung von Antwortformularen durch den Benutzer zulassen	Benutzer können in der Domäne neue Antwortformulare registrieren und vorhandene ändern.
	Löschen von Antwortformularen durch den Benutzer zulassen	Benutzer können in der Domäne Antwortformulare der Kunden löschen.
	Bearbeitung der Kategoriezugehörigkeit durch den Benutzer zulassen	Benutzer können in der Domäne die Zugehörigkeit von Kunden zu Kategorien hinzufügen und ändern.
	Löschen der Kategoriezugehörigkeit durch den Benutzer zulassen	Benutzer können in der Domäne die Zugehörigkeit von Kunden zu Kategorien entfernen.
	Bearbeitung der Dialogteilnahme durch den Benutzer zulassen (Web)	Benutzer können in der Domäne Kunden als Dialogteilnehmer hinzufügen. Dies ist nur für Dialoge möglich, deren Gruppen das manuelle Hinzufügen von Teilnehmern erlauben.
	Löschen der Dialogteilnahme durch den Benutzer zulassen	Benutzer können in der Domäne Kunden aus der Teilnahme an einem Dialog löschen.
Dialogfeld	Bearbeiten eines Dialogs	Benutzer können den Dialog in Visual Dialogue bearbeiten.

	Dialog löschen	Benutzer können den Dialog in Visual Dialogue löschen.
	Dialog ausführen	Benutzer können den Dialog in der *Ausführungsansicht* in Visual Dialogue ausführen.
	Dialog leeren	Benutzer können den Dialog in Visual Dialogue leeren. Dies entfernt alle Ausführungsdaten des Dialogs, einschließlich der Teilnehmer.
Berichtsvorlage	Berichtsvorlage bearbeiten	Benutzer können in Visual Dialogue die Berichtsvorlage bearbeiten und löschen.
	Bericht ausführen	Benutzer können den Bericht im Report Portal ausführen.
	Archivierte Versionen des Berichts anzeigen	Benutzer können archivierte Berichte im Report Portal ansehen.
	Speichern von Berichten im Archiv durch den Benutzer zulassen	Benutzer können von ihnen in Report Portal ausgeführte Berichts im Berichtsarchiv speichern.
	Berichte aus Archiv löschen	Benutzer können archivierte Berichts aus dem Berichtsarchiv in Report Portal löschen.
	Systemparameter überschreiben	Benutzer können die Parameter „Benutzer-ID“ und „Benutzername“ in Systemberichten ändern.
Telemarketing-Projekt	Als Telemarketing-Betreiber agieren	Benutzer können als TM-Betreiber in Telemarketing-Projekten agieren. Dieses Zugriffsrecht kann in Visual Dialogue im Telemarketing-Designer zugewiesen werden.
	Zugriff des Benutzers auf Telemarketing-Projekte zulassen	Benutzer können im Webmodul „Telemarketing“ die Berichtsseite der Projekte anzeigen.

Benutzersitzungsprotokoll

Benutzersitzungsprotokoll bietet Zugriff auf den Verlauf der Benutzeranmeldungen, inklusive die gegenwärtig angemeldeten Benutzer.

Hinweis: Um einen aktiven Benutzer abzumelden, wählen Sie den Benutzer im Knoten „Aktiv“, und wählen Sie anschließend im Menü die Optionen **Element** > **Sitzung abmelden**.

Allgemeine Verwaltung

Allgemeine Verwaltung ist eine Sammlung verschiedener konfigurierbarer Elemente.

Parametersammlungen

Parametersammlungen sind diverse Gruppen von *Systemparametern*. Ein Systemparameter ist eine Einstellung, die nur von Systemadministratoren geändert werden kann.

Es ist möglich, Parametersammlungen und einzelne Parameter hinzuzufügen.

Anwendungssysteme

Ein *Anwendungssystem* bezeichnet entweder ein standardmäßiges Anwendungsmodul von Portrait Dialogue oder ein anderes System, das in Dialogue Server integriert ist. Das Konzept von Anwendungssystemen wird in folgenden Kontexten verwendet:

- **Anmelden**

  Wenn sich eine Anwendung am Dialogue Server anmeldet, muss sie ein gültiges Anwendungssystem angeben.

- **Ereignisse**

  Wenn Dialogue Server ein Ereignis mitgeteilt wird, verfolgt der Server im Weiteren das Anwendungssystem, von dem das Ereignis stammt.

Tabellenwartung

Tabellenwartung gibt dem Benutzer Zugriff auf Datenbanktabellen in Dialogue Database.

- **Eigenschaften**

  Wählen Sie für eine beschreibende Liste einer Tabellenspalte die Tabelle aus, und wählen Sie anschließend die Optionen **Element** > **Eigenschaften**.

- **Öffnen**

  Wählen Sie zum Bearbeiten von Daten erst die Tabelle aus und dann die Optionen **Element** > **Öffnen**.

Warnung: Die Bearbeitung von Daten in einer Datenbanktabelle sollte ausschließlich von zertifizierten Technikern mit Kenntnis des Dialogue Database-Modells vorgenommen werden.

Test-Dienstprogramme

Der Microsoft-Support verwendet in der Regel drei Hauptdienstprogramme zum Debuggen von MSDTC-Transaktionen und dazugehörigen Fehlern:

- **DTCPing** – Download und Dokumentation unter **http://support.microsoft.com/default.aspx?scid=kb;en-us;306843**. Verwenden Sie das DTCPing-Tool, um zu prüfen,

ob Firewalls oder Netzwerke verteilte Transaktionen unterstützen. Das DTCPing-Tool muss sowohl auf dem Client- als auch auf dem Servercomputer installiert werden und ist eine gute Alternative zum Dienstprogramm DTCTester, wenn auf keinem der Computer SQL Server installiert ist.

- **DTCTester** – Download und Dokumentation unter **http://support.microsoft.com/default.aspx?scid=kb;en-us;293799**. Verwenden Sie das DTCTester-Tool, um zu prüfen, ob Firewalls oder Netzwerke verteilte Transaktionen unterstützen. Das Dienstprogramm DTCTester verwendet ODBC, um die Unterstützung von Transaktionen in einer SQL Server-Datenbank zu prüfen. Daher muss SQL Server auf einem der zu prüfenden Computer installiert sein.
- **NetMon** – Befindet sich auf dem Windows-Installationsdatenträger oder im Resource Kit.

Kapitel

2

Kundendomänen und -ausdrücke

In diesem Abschnitt:

- Kundendomänen und -ausdrücke 36
- Erste Schritte mit Domänen 36
- Kundendomänen in Visual Dialogue 40
- Kundendomänen in Customer View 40
- Details zur Konfiguration einer Kundendomäne 42
- Kundendomänenüberprüfung 52
- Formeln 58

Kundendomänen und -ausdrücke

Einleitung

Portrait Dialogue kann Kundendaten aus einer Vielzahl von Quellen aufrufen. Zur Definition des Zugriffs auf Kundendaten wird das Konzept der *Kundendomäne* angewendet.

Kundendomänen werden in Dialogue Admin eingerichtet und repräsentieren alle unterschiedlichen Kundentypen in Dialogue Server. Beispiele für Kundendomänen sind: Personen, B2B-Markt, Haushalte oder potentielle Kunden. Eine Kundendomäne kann sich aus mehr als einer Datenquelle zusammensetzen.

Dialogue Server verwendet die Kundendomänendefinition, um auf Kundendaten zuzugreifen und diese Daten über die API anderen Anwendungen zugänglich zu machen.

Den Kundendomänen übergeordnet wird eine *Ausdruckssprache* definiert. **Ausdrücke** sind ein vereinheitlichter Weg, Kundendaten abzufragen. Dies ist von der zugrundeliegenden Informationsquelle unabhängig.

Erste Schritte mit Domänen

Erste Schritte mit Domänen bietet eine Einführung in die Konfiguration von Domänen. Dabei wird in einem Schritt-für-Schritt-Verfahren eine einfache Domäne erstellt.

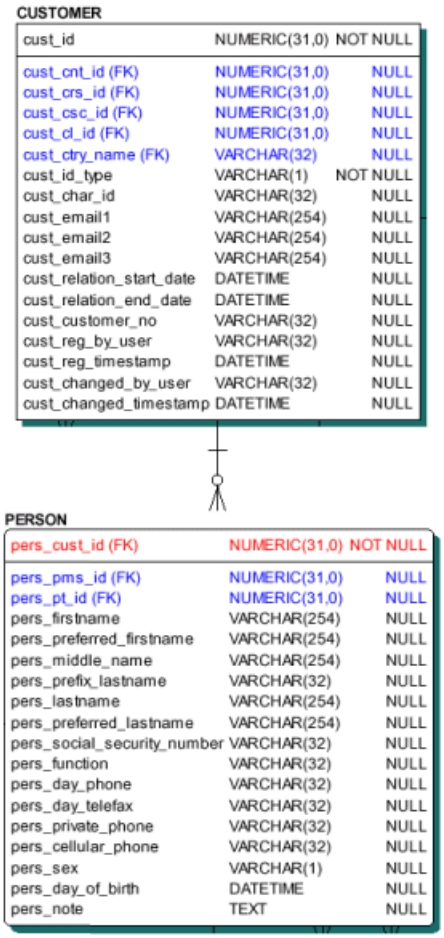
Die Domäne, die mit diesem Tutorial erstellt wird, repräsentiert Personen in der Standardkundendatenbank.

Der erste Schritt in diesem Tutorial ist das **Erstellen der SQL-Hauptgruppe**.

Schritt 1 – Erstellen der SQL-Hauptgruppe

Wir müssen zur Definition einer Kundendomäne zunächst eine SQL-Anweisung schreiben, welche die Kunden aus der Datenbank auswählt. Wir nennen dies die *SQL-Hauptgruppe*, da sie die Hauptdatengruppe in der Domäne definiert.

In diesem Beispiel wählen wir Einträge aus der Standardkundendatenbank aus. Dies erfordert die Auswahl aus zwei Tabellen: *Kunde* und *Person*. Wie in der unteren Abbildung dargestellt, wird eine SELECT-Anweisung geschrieben, um die passende Spaltenauswahl aus den beiden Tabellen abzurufen.




```
select
  cust_char_id,
  cust_id,
  cust_reg_by_user,
  cust_reg_timestamp,
  cust_changed_by_user,
  cust_changed_timestamp,
  cust_email1,
  cust_email2,
  pers_firstname,
  pers_lastname,
  pers_day_phone,
  pers_private_phone,
  pers_day_telefax,
  pers_cellular_phone,
  pers_sex,
  pers_day_of_birth,
  pers_pms_id,
  pers_note,
  pers_firstname + ' ' + pers_lastname as fullname
from mhcust.customer, mhcust.person
where cust_id = pers_cust_id
```

Die SQL-Anweisung wird im SQL-Repository in Dialogue Admin gespeichert. Wählen Sie zum Hinzufügen einer SQL-Definition zum SQL-Repository eine SQL-Gruppe im Browser auf der linken Seite aus, und wählen Sie im Menü die Optionen **Element** > **Neu...**.

Schritt 2 – Hinzufügen der Domäne

Im zweiten Schritt wird die Domäne selbst hinzugefügt. Markieren Sie in Dialogue Admin den Knoten *Kundendomänen*, und wählen Sie im Menü die Optionen **Element** > **Neu...**. Daraufhin wird ein Fenster angezeigt, in dem Sie nach dem Namen der neuen Domäne gefragt werden.

Nach der Angabe des Namens wird in Dialogue Admin der *Kundendomänen-Editor* angezeigt.

Schritt 3 – Hinzufügen der Hauptdatengruppe

Im nächsten Schritt wird die Hauptdatengruppe mit der SQL-Anweisung aus Schritt 1 zur Domäne hinzugefügt. Wählen Sie im Menü die Optionen **Gruppe** > **Neu...**, um eine neue Gruppe hinzuzufügen. Daraufhin wird das Fenster *Gruppeneigenschaften* angezeigt.

Wählen Sie zuerst den Namen der Hauptgruppe und optional eine Beschreibung.

Wählen Sie als nächstes die Registerkarte *Datenquelle*, um die in Schritt 1 erstellte SQL-Anweisung zur Gruppe hinzuzufügen.

Verwenden Sie zum Auswählen der Anweisung die Schaltfläche „...“. Daraufhin wird das SQL-Browserfenster geöffnet.

Suchen Sie die in Schritt 1 erstellte SQL-Anweisung, und klicken Sie auf *OK*. Jetzt ist die Gruppe zum Abruf von Daten mit dem angegebenen SELECT-Befehl eingerichtet.

Schließen Sie das Hinzufügen der Gruppe ab, indem Sie im Fenster „Eigenschaften“ auf *OK* klicken.

Schritt 4 – Hinzufügen der Felder

Nachdem wir die Hauptdatengruppe hinzugefügt haben, müssen nun die Felder zur Gruppe hinzugefügt werden. Wählen aus dem Menü die Optionen **Feld** > **Felderliste aktualisieren...**. Die Spalten des SELECT-Befehls erscheinen nun als Felder der Gruppe.

Um Kunden in der Domäne eindeutig identifizieren zu können, muss dem System mitgeteilt werden, welches Feld diese eindeutige Identifizierung leistet. Daher müssen wir ein Feld angeben, welches das identifizierende Feld der Gruppe ist. Die Spalte *cust_char_id* in der *Kundendatenbank* ist als eindeutige ID einer Domäne konzipiert. Um die eindeutige ID anzugeben, markieren Sie *cust_char_id*, und wählen Sie aus dem Menü die Optionen **Feld** > **Als ID verwenden**. *cust_char_id* ist nun als identifizierendes Feld gekennzeichnet und ein Schlüsselsymbol wird vor dem Feldnamen angezeigt.

Schritt 5 – Bearbeiten von Feldeigenschaften

Ein Feld verfügt über viele konfigurierbare Eigenschaften. Alle Eigenschaften werden detailliert im Abschnitt **Felder** auf Seite 47 beschrieben.

Im Fenster „Eigenschaften“ auf der Registerkarte *Allgemein* können Sie dem Feld einen benutzerfreundlichen Namen geben, der sich vom Spaltennamen in der Datenbank unterscheidet. Außerdem können Sie eine ausführlichere Feldbeschreibung hinzufügen, um Benutzern zu helfen, das Feld zu verstehen.

Im Fenster „Eigenschaften“ auf der Registerkarte *Web* können Sie Optionen einrichten, die mit Webanwendungen zu tun haben, wie beispielsweise ob ein Feld in Customer View suchbar sein soll.

Schritt 6 – Aktivieren der neuen Domäne

Die neue Domäne ist nun beinahe testbereit. Allerdings muss sie zunächst aktiviert und gespeichert werden.

Einige Schlüsseleigenschaften der Domäne werden im Fenster der Domäneneigenschaften gesteuert. Wählen Sie aus dem Menü **Datei** > **Eigenschaften...** um dieses Fenster zu öffnen.

Aktivieren Sie die Domäne, indem Sie im Eigenschaftsfenster *Aktiviert* anklicken. Eine Domäne ist standardmäßig deaktiviert, weil sie im Aufbau nicht für die Endbenutzer der Webanwendungen sichtbar sein sollte.

Klicken Sie im Eigenschaftenfenster auf die Registerkarte *Internet*, um die Domäne in Customer View verfügbar zu machen. Aktivieren Sie *In Customer View anzeigen*. Die Domäne ist nun in Customer View auf der Suchseite verfügbar.

Die Registerkarte *Internet* enthält noch einige weitere Optionen.

Hinweisfeld ist ein optionales Feld, das in Customer View auf einer getrennten Registerkarte angezeigt wird.

Adressgruppe ist eine optionale Gruppe, die Adressen enthält. Falls festgelegt, wird sie in der Benutzerschnittstelle von Customer View speziell behandelt.

Internetprofil des Kunden aktivieren legt fest, ob die Anwendung „Customer Web Access“ die Kundenprofilseiten für diese Domäne aktiviert. Falls aktiviert, können E-Mails, die Verknüpfungen zur Profilseite enthalten, an einen Kunden in dieser Domäne versendet werden.

Anzeigefeld gibt das anzuzeigende Feld an, wenn Verknüpfungen zu Kunden erstellt werden, z. B. in Customer View in der Liste der *letzten Kunden*. Standardmäßig wird die Kunden-ID verwendet. Für einen benutzerfreundlicheren Anzeigenamen verwendet diese Anleitung aber eine Verkettung aus Vor- und Nachname.

Abschließend kann das Fenster der Domäneneigenschaften geschlossen und aus dem Menü **Datei** > **Speichern** ausgewählt werden, um alle Änderungen der neuen Domäne zu speichern.

Schritt 7 – Testen der Domäne in Customer View

Öffnen Sie Customer View, und wählen Sie aus dem linken Menü die Optionen **Suchen** > **Personen**, um nach Kunden in der neuen Domäne zu suchen. Felder aus der Domäne sind nun als Suchkriterien verfügbar. Beachten Sie, dass die neu bearbeiteten Feldnamen angezeigt werden, und nicht die aus SQL abgerufenen Namen.

Führen Sie eine Suche aus, indem Sie einige Suchkriterien eingeben (z. B. A* im Feld „Vorname“). Doppelklicken Sie in der Liste *Suchergebnis* auf einen einzelnen Kunden, um ihn anzuzeigen.

Schritt 8 – Testen der Domäne im Auswahldesigner

Starten Sie Visual Dialogue, um die neue Domäne im Auswahldesigner zu testen. Wählen Sie aus dem Menü **Datei** > **Neu** > **Auswahl**. Der *Assistent für neue Auswahl* wird angezeigt, und die neue Kundendomäne kann in der Dropdownliste *Kundendomäne* ausgewählt werden.

Der Auswahldesigner wird nun ohne irgendwelche definierten Kriterien angezeigt.

Auf der rechten Seite des Bildschirms zeigt die *Toolbox von Visual Dialogue* Felder der neuen Domäne an. Die Toolbox hilft dem Benutzer, Kriterien für Auswahlen zu definieren.

Beispiel: Doppelklicken Sie in der Toolbox auf *Registriert*, um ein Kriterium zu erstellen, das mit dem Feld *Registriert* verbunden ist. Daraufhin wird ein Kriteriumsfenster angezeigt.

Nachdem das Kriterium definiert wurde, können Sie **Auswahl** > **Musterauswahl** verwenden, um eine Vorschau der Kunden zu erhalten, die das Kriterium erfüllen.

Schritt 9 – Weitere Informationen zu Domänen

Detaillierte Informationen zu Kundendomänen finden Sie im Abschnitt **Details zur Konfiguration einer Kundendomäne**. Dort werden alle Eigenschaften einer Domäne selbst, ihrer Gruppen sowie ihrer Felder beschrieben.

Kundendomänen in Visual Dialogue

Die verschiedenen Module innerhalb von Visual Dialogue verwenden Kundendomänendefinitionen, um den Benutzer Auswahlen, Dialoge und Nachrichtenvorlagen entwerfen zu lassen.

Kundendomäne und Dialoge

Ein Dialog ist immer mit einer spezifischen Kundendomäne verbunden. Dies bedeutet, alle Teilnehmer dieses Dialogs gehören zu dieser Kundendomäne. Diese Einschränkung ist nötig, da Operationen in einem Dialog basierend auf der Definition einer Domäne eingerichtet werden.

Wenn ein Dialog ausgeführt wird, werden Kundendaten aus der Domäne vielfältig verwendet. Eine E-Mail-Operation ruft beispielsweise Daten durch die Kundendomäne ab, wenn der Inhalt der E-Mail erstellt wird. Eine Aufteilungsoperation verwendet Kundendaten, um zu entscheiden, wie Kunden in verschiedene Gruppen aufgeteilt werden.

Wenn Sie die Teilnehmer einer Dialoggruppe anzeigen, werden die angezeigten Information unter Verwendung der Kundendomäne abgerufen.

Kundendomäne und Auswahlen

Wenn Auswahlen in Visual Dialogue entworfen werden, kann der Benutzer Kriterien erstellen, die auf Informationen basieren, die durch die Kundendomäne verfügbar sind. Auswahlkriterien werden definiert, indem die Ausdruckssprache für Adressgruppen und -felder in der Domäne verwendet wird. Eine Auswahl ist immer mit einer bestimmten Kundendomäne verbunden.

Wenn Kunden in einer Auswahl angezeigt werden (Musterauswahl), sind die angezeigten Informationen eine Teilmenge aller Felder der Domäne.

Kundendomäne und Nachrichten

Wenn eine neue Nachrichtenvorlage erstellt wird (E-Mail, SMS, Exportieren usw.), muss der Benutzer die zu adressierende Kundendomäne auswählen. Wenn die Vorlage entworfen wird, kann der Benutzer Felder aus der Domäne auswählen, die in die Vorlage eingefügt werden. Die Empfängeradresse einer E-Mail kann beispielsweise als die E-Mail-Adresse der Person in der Domäne festgelegt werden. Darüber hinaus können Kundennamen und weitere Informationen einbezogen werden, um die Kommunikation mit dem Kunden zu personalisieren.

Kundendomänen in Customer View

Customer View ist eine kundenorientierte Webanwendung, die verwendet wird, um Kundeninformationen zu finden, anzuzeigen und zu bearbeiten. Sie basiert stark auf dem Konzept der Kundendomäne. Die in Customer View zur Verfügung stehenden Kundendaten sind die Gruppen und Felder, die in Ihren Kundendomänen definiert sind.

Suchen nach Kunden

Alle Felder in einer Kundendomäne können als suchbar in Customer View eingerichtet werden. Meistens ist es angebracht kleine Sätze von Feldern auf der Suchseite verfügbar zu haben. Dies kann in Dialogue Admin konfiguriert werden.

Wenn der Benutzer mit einem oder mehreren Kriterien sucht, erstellt die Webanwendung im Hintergrund einen Suchausdruck, der an Dialogue Server geschickt wird. Ein Satz von Kundendaten aus der Domäne, wird dann der Webanwendung als XML-Dokument zurückgegeben.

Anzeigen eines Kunden

Wenn ein spezifischer Kunde in Customer View geöffnet wird, fragt die Webanwendung am Server verfügbare Daten in der Domäne ab. Die Kundenseite ist dynamisch aufgebaut und basiert auf der Konfiguration der Kundendomäne.

Registrieren und Bearbeiten eines Kunden

Die Webseite, die verwendet wird, um Kundendaten zu registrieren und zu aktualisieren, ist durch Verwendung der Domänendefinition dynamisch aufgebaut. Die Gruppen oder Felder einer Domäne können in Dialogue Admin aktualisierbar eingerichtet werden.

Änderungen werden als XML-Dokument an Dialogue Server gesendet, wenn sie angewendet werden sollen. Der Server wird die Daten dann validieren und die notwendigen Schritte zum Übernehmen der Änderungen durchführen. Alles erfolgt gemäß der aktuellen Domänenkonfiguration.

In der Domäne definierte Feldvalidierungsregeln werden dynamisch als JavaScript auf der Webseite „Kunde bearbeiten“ implementiert.

Suchfelder in der Domäne können als Kombinationsfelder angezeigt werden, wenn ein Kunde bearbeitet wird.

Zusätzlich können Regeln für Suchfeldwerte definiert werden, wenn ein anderes Feld einen Wert erhält. Beispielsweise kann nach der *Stadt* gesucht werden, wenn der Benutzer die *Postleitzahl* eingibt. Solche Suchen werden in der Domäne definiert und als Webdienstaufrufe in JavaScript implementiert.

Weitere Webanwendungen

Zu weiteren Webanwendungen gehören *Telemarketing* und *Customer Web Access*. Ebenso wie Customer View verwenden diese Anwendungen die Domänendefinition, um dynamisch Seiten für das Anzeigen und Bearbeiten von Kundendaten zu erstellen.

Erstellen eigener Webanwendungen

Für das Erstellen von Webanwendungen in Microsoft ASP.NET ist ein Framework vorhanden. Dieses Framework heißt *Web Framework* und ist vollständig auf der Installations-CD dokumentiert.

„Web Framework“ ist eine Klassenbibliothek, die erstellt wurde, um allgemeine Dienste bereitzustellen. Diese Dienste werden von Webanwendungen benötigt, die mit Dialogue Server kommunizieren. Das Framework erleichtert das Schreiben von Anwendungen, die Kundendaten lesen und aktualisieren.

Alle Standardwebanwendungen verwenden Web Framework.

Details zur Konfiguration einer Kundendomäne

Kundendomänen werden in Dialogue Admin definiert.

Markieren Sie im Admin-Browser den Knoten *Kundendomänen*, und wählen Sie aus dem Menü die Optionen **Element** > **Neu**, um eine neue Kundendomäne zu erstellen. Markieren Sie eine bestehende Domäne im Admin-Browser, und wählen Sie aus dem Menü die Optionen **Element** > **Öffnen**, um diese zu bearbeiten.

Das Fenster zum Bearbeiten der Kundendomäne ist unten dargestellt. Die linke Seite ist eine Gliederung, welche die Datengruppen der Domäne enthält. Der oberste Knoten ist die Hauptgruppe, die alle Kunden in der Domäne definiert. Die Untergruppen enthalten zusätzliche Informationen wie Unternehmen, Adressen, Aktivitäten usw.

Der rechte Teil des Fensters verfügt über zwei Modi: *Datenfelder* und *Parameterbindungen*. Der Modus wird mittels der unteren Registerkarten geändert. Der Modus „Datenfelder“ zeigt alle Felder, die in der links ausgewählten Datengruppe definiert sind. Diese Felder besitzen einen Datentyp, eine Größe und viele andere Eigenschaften. Der Modus „Parameterbindungen“ beschreibt, wie Felder in einer Untergruppe mit der Hauptgruppe verbunden sind.

Eigenschaften von Kundendomänen

Eine Kundendomäne verfügt über einen Satz von Eigenschaften. Wählen Sie das Menü **Datei** > **Eigenschaften**, um Eigenschaften anzuzeigen oder zu bearbeiten. Das Eigenschaftenfenster ist in mehrere Registerkarten unterteilt:

--> **Allgemeine Eigenschaften**

--> **Webeigenschaften**

--> **Anmeldeeigenschaften**

Allgemeine Eigenschaften

Die erste Registerkarte des Eigenschaftenfensters enthält die Schlüsseldomäneninformationen. *Stammverzeichnis* ist der Pfad, unter dem Dateien gespeichert werden, die mit der Domäne verbunden sind. Dies können Vorlagen, Nachrichten und Protokolle sein. Normalerweise wird dieser Pfad vom Benutzer nicht verändert, da er bei der Erstellung einer Domäne automatisch generiert wird.

Wenn *aktiviert* nicht ausgewählt ist, wird die Domäne deaktiviert, und ist in Client-Anwendungen (Customer View, Visual Dialogue usw.) nicht verfügbar.

Webeigenschaften

Die zweite Registerkarte des Eigenschaftenfensters enthält Einstellungen, die wichtig für Webanwendungen sind.

Aktivieren Sie *In Customer View anzeigen*, um die Domäne in Customer View verfügbar zu machen.

Hinweisfeld ist ein optionales Feld, das in Customer View auf der Registerkarte „Hinweis“ der Kundenseite angezeigt wird.

Adressgruppe ist eine Datengruppe, die Adressinformationen enthält. Falls eingestellt, wird diese Gruppe passender in Customer View angezeigt.

Prüfung auf Duplikate aktivieren bestimmt, ob die Prüfung auf Duplikate für die Domäne aktiviert ist. Die Prüfung auf Duplikate wird verwendet, wenn Kunden in Customer View eingefügt oder bearbeitet werden. Wählen Sie zum Verwenden der Prüfung auf Duplikate die Option *Prüfung auf Duplikate aktivieren*, und implementieren Sie in der Hauptdatengruppe der Domäne im Aktualisierungs-Plug-In die Methode *CheckForDublicates(...)*. Details zur Implementierung von Aktualisierungs-Plug-Ins finden Sie unter **Kundendaten-Plug-Ins**.

Internetprofil des Kunden aktivieren legt fest, ob die Anwendung „Customer Web Access“ die Kundenprofilseiten für diese Domäne aktiviert. Falls aktiviert, können E-Mails, die Verknüpfungen zur Profilseite enthalten, an einen Kunden in dieser Domäne versendet werden.

Anzeigefeld ist das in Beschriftungen und Verknüpfungen angezeigte Domänenfeld, wenn es sich auf einen Kunden bezieht. Wird kein Anzeigefeld angegeben, wird die Kunden-ID verwendet.

Anmeldeeigenschaften

Die dritte Registerkarte des Eigenschaftsfensters enthält Einstellungen, die wichtig für die integrierte Funktion der Kundenauthentifizierung sind.

Wählen Sie *Kundenanmeldung aktivieren*, um Kundenanmeldungen für die aktuelle Kundendomäne zu aktivieren.

Die Kundenanmeldung aktiviert die Authentifizierung von Kunden in der Anwendung „Customer Web Access“. Zum Beispiel kann ein Kunde zur Anmeldung aufgefordert werden, bevor er einen Fragebogen ausfüllt. Nach der Anmeldung ist der Kunde identifiziert, und das Antwortformular wird mit dem Kunden verbunden.

Außerdem kann die Anmeldefunktion in benutzerdefinierten Anwendungen verwendet werden, da die Methoden zur Kundenauthentifizierung in der Dialogue Server-API verfügbar sind.

Eine *Anmelde-ID* muss ausgewählt sein. Diese muss ein eindeutiges Feld in der Kundendomäne sein. Typischerweise ist die Kontakt-ID (wie im unteren Screenshot) eindeutig, der Vorname dagegen nicht.

Kennwort kann so eingerichtet werden, dass ein Kennwort entweder automatisch von Dialogue Server generiert wird, oder ein Domänenfeld dafür ausgewählt werden kann.

Die Einstellung *Kennwort für Webprofilanmeldung nicht erforderlich* legt fest, ob nur die *Anmelde-ID* (und nicht beides, *Anmelde-ID* und Kennwort) angegeben werden muss, wenn ein Kunde sich im Webprofil anmeldet. Sie können dieses Verhalten in Visual Dialogue für Fragebögen auf einer Pro-Fragebogen-Grundlage steuern.

Datengruppen

Verschiedene Gruppentypen

Eine Domäne enthält immer eine Hauptgruppe von Kundendaten. Diese Hauptgruppe sollte ihre Daten immer durch Verwendung einer SQL-SELECT-Anweisung abrufen, die einen Datensatz mit genau *einer Zeile pro Kunde* zurückgibt. Eine Spalte des zurückgegebenen Datensatzes identifiziert den Kunden, auch genannt *mh_customer_id*.

Darüber hinaus kann eine Kundendomäne Untergruppen von Kundendaten enthalten. Diese Datengruppen sind verschiedenen Typs:

- **1:n-Gruppen (gibt mehrere Zeilen zurück)**

  Diese Gruppen geben *jede beliebige* Anzahl von Zeilen pro Kunde zurück. Beispiel: Alle Mitarbeiter eines Unternehmens. (Die Hauptgruppe enthält dann alle Unternehmen).

- **1:1-Gruppen**

  Diese Gruppen geben *eine oder keine* Zeile pro Kunde zurück. Beispiel: Die Hauptadresse des Kunden.

- **Boolesche Gruppentypen**

  Diese Gruppen definieren ein boolesches Datenfeld, das entweder *True* (Wahr) oder *False* (Falsch) für einen gegebenen Kunden zurückgibt. Beispiel: Erweitertes SQL kann dazu verwendet werden, zu bestimmen, ob ein Kunde für den Kauf von Kreditkarten qualifiziert ist.

Hinweis: Die Domänendaten sollten über dieselbe Datenbankverbindung (Verbindungszeichenfolge) verfügbar sein wie Dialogue Database, um die beste Leistung beim Arbeiten mit einer großen Anzahl von Kunden zu erzielen.

Systemgruppen

Systemgruppen sind spezielle Gruppen, die vom System selbst definiert werden. Informationen zu Systemgruppen finden Sie im Abschnitt **Systemgruppen**.

Hinzufügen einer neuen Gruppe

Klicken Sie im Domänenfenster auf der linken Seite auf die Gruppengliederung, und wählen Sie aus dem Menü **Gruppe** > **Neu...**, um eine neue Datengruppe zu einer Domäne hinzuzufügen. Das Fenster „Gruppeneigenschaften“ wird angezeigt.

Das Fenster „Gruppeneigenschaften“

Das Fenster der Datengruppeneigenschaften enthält Einstellungen, die den Gruppentyp, die Datenquelle, die Aktualisierungsmethode und Webeinstellungen definieren. Das Eigenschaftenfenster ist in mehrere Registerkarten unterteilt:

--> **Allgemeine Eigenschaften**

--> **Datenquelleneigenschaften**

--> **Datenaktualisierungseigenschaften**

--> **Webeigenschaften**

Allgemeine Eigenschaften

Die erste Registerkarte enthält Schlüsseleigenschaften. *Gruppenname* ist der Name der Gruppe. Der Name kann aus Klein- und Großbuchstaben, Zahlen und dem Unterstrich bestehen. Der Name muss mit einem Buchstaben beginnen. Er darf **keine** Leerzeichen enthalten. Namen von **Ausdrucksfunktionen** sind reserviert und können nicht als Gruppennamen verwendet werden. Außerdem muss der Name konform sein mit der Namenssyntax der **Extensible Markup Language (XML) 1.0 (Vierte Edition)**. Sie müssen die Funktion „Kundendomäne überprüfen“ verwenden, um den Namen zu validieren.

Nur für fortgeschrittene Anwendung muss aktiviert sein, wenn die Gruppe normalerweise für den Benutzer nicht sichtbar ist.

Boolescher Gruppentyp bestimmt, ob diese Gruppe vom booleschen Typ ist.

1:n-Gruppentyp gibt an, ob diese Gruppe mehrere Datensätze pro Kunde zurückgibt.

Standardzeilenausdruck (optional) gibt einen Ausdruck zum Abrufen der Standardzeile für eine 1:n-Gruppe an (momentan nicht verwendet).

Identifizierendes Feld ist der eindeutige Schlüssel für die Gruppe. Es ist optional für alle Gruppen außer der Hauptgruppe. In der Hauptgruppe stellt dies die Kunden-ID dar („mh_customer_id“). *Identifizierendes Feld* kann auch aus dem Feldmenü festgelegt werden, wenn ein Feld ausgewählt ist.

Datenquelleneigenschaften

Die Registerkarte „Datenquelle“ lässt den Benutzer die Datenquelle auswählen. Dies geschieht, indem eine SQL-Anweisung im SQL-Repository ausgewählt wird. Wenn eine neue Gruppe erstellt wird, definiert der Benutzer die SQL-Anweisung normalerweise im Voraus.

Die SQL-Anweisung sollte allgemein die Spalten auswählen, die den Feldern in der Gruppe entsprechen. Dennoch gibt es verschiedene Regeln für die Anwendung der SQL-Anweisungen, abhängig vom Gruppentyp.

- **Hauptgruppe**

  Die SQL-Anweisungen der Hauptgruppe müssen das Kunden-ID-Feld zurückgeben. Beachten Sie, dass dieses ID-Feld eine Zeichenfolgenspalte sein sollte.

- **1:1-Gruppen**

  Die SQL-Abfragen einer 1:1-Gruppe dürfen nur *eine oder keine* Zeile pro Kunde zurückgeben. Wenn mehr als eine Zeile zurückgegeben wird, funktioniert die Domäne nicht ordnungsgemäß.

- **1:n-Gruppen**

  SQL-Abfragen bei 1:n-Gruppen können eine beliebige Anzahl von Zeilen pro Kunde zurückgeben.

- **Boolesche Gruppentypen**

  Eine boolesche Gruppe stellt einen Wert dar, der einen der folgenden Status erfüllt: *true* (wahr) oder *false (falsch)*. Die SQL-Anweisungen boolescher Gruppen sollten *eine* Zeile zurückgeben, wenn das von der Gruppe definierte boolesche Feld mit *true* (wahr) bewertet werden soll. Wird *keine* Zeile zurückgegeben, wird sie mit „false“ (falsch) bewertet. Die SQL-Anweisungen sollten niemals mehrere Zeilen pro Kunde zurückgeben.

Die Option *Datenabfrage in separaten Anfragen* betrifft die Art, wie Dialogue Server SQL-Anfragen zur Abfrage von Daten aus Domänengruppen erstellt. Falls aktiviert, wird der Server versuchen, auf die Domänengruppe durch Verwendung separater Anfragen zuzugreifen, anstatt die ausgewählte SQL mit SQL-Anweisungen von anderen Domänengruppen zu verbinden. Ist diese Option nicht aktiviert, wird der Server vorzugsweise externe Verknüpfungen für die SQL der Hauptgruppe in der Domäne verwenden. Beachten Sie, dass diese Option nur für 1:1-Gruppen gilt.

Nachdem die Datenquelle angegeben wurde, kann der Benutzer auf *OK* klicken und **Feld** > **Feldliste aktualisieren** auswählen. Das Domänenfenster zeigt dann alle verfügbaren Felder aus der SQL an.

Die SQL-Anweisungen können keine Host-Variablen enthalten. Dennoch müssen Spalten, die sich auf Daten in der Hauptgruppe beziehen, in den Select-Teil mit eingeschlossen und als Parameterbindungen definiert werden.

Wählen Sie aus dem Hauptmenü **Parameter** > **Neu...**, um einen neuen Parameter hinzuzufügen. Daraufhin wird ein Fenster angezeigt, in dem der Benutzer ein Feld aus der neuen Datengruppe auswählen kann, und dieses Feld an ein Feld der Hauptgruppe oder einen konstanten Wert binden kann.

Der grau hinterlegte Teil der Registerkarte „Datenquelle“ ist in dieser Version deaktiviert und für den zukünftigen Gebrauch gedacht.

Datenaktualisierungseigenschaften

Eine Datengruppe kann optional aktualisierbar sein. Es werden zwei Methoden zur Datenaktualisierung unterstützt:

1) *Automatische Aktualisierung* unterstützt nur die Aktualisierung einer Datenbanktabelle. Mithilfe dieser Methode generiert Dialogue Server SQL-Anweisungen, um Aktualisierungen anzuwenden. Der Name der zu aktualisierenden Tabelle muss angegeben werden. Optional kann Dialogue Server eindeutige IDs generieren, wenn Datensätze eingefügt werden. Das als identifizierend angegebene Feld (*Registerkarte „Allgemeine Einstellungen“*) erhält die generierte ID. Sie müssen einen eindeutigen Namen für die Sequenz angeben (z. B. MY_SEQ_PERSONS), um diese Funktion zu verwenden.

2) Das *Verwenden eines Plug-Ins zur Datenaktualisierung* ist die flexibelste Aktualisierungsmethode. Der Benutzer gibt dann ein Plug-In an, in dem der Aktualisierungsprozess als Skript implementiert ist. Das Plug-In muss im *Plug-In-Repository definiert sein.* Details zur Implementierung von Aktualisierungs-Plug-Ins finden Sie unter **Kundendaten-Plug-Ins**.

Webeigenschaften

In Customer View anzeigen gibt an, ob die Datengruppe angezeigt wird, wenn in Customer View ein Kunde geöffnet wird. Das Deaktivieren dieser Eigenschaft überlagert die ähnliche Einstellung eines Datenfeldes.

Daten bei Bedarf laden legt fest, ob Customer View Daten aus dieser Datengruppe abruft, wenn der Benutzer auf die Registerkarte klickt, welche die Liste der Daten dieser Gruppe enthält. Anderenfalls werden die Daten abgerufen, wenn eine Kundenseite geöffnet wird. Diese Einstellung wird für Datengruppen angewendet, die in Customer View im unteren Teil (die Registeransicht) der Kundenseite angezeigt werden.

In Customer View suchbar legt fest, ob Felder dieser Datengruppe in Customer View als Suchfelder angezeigt werden können. Das Deaktivieren dieser Eigenschaft überlagert die ähnliche Einstellung eines Datenfeldes.

Bearbeitungsmodus in Registerform verwenden gilt nur für 1:n-Gruppen. Durch Aktivieren dieser Option wird die Gruppe in Customer View im oberen Teil der Kundenseite angezeigt. Das Wechseln zwischen Datenzeilen in der Gruppe kann dann durch eine Dropdownliste erfolgen. Der in der Dropdownliste angezeigte Wert entspricht dem im *Anzeigefeld der Registerkarte* angegebenen Wert. Diese Option ist zum Beispiel geeignet, wenn der Kunde mehrere Adressen hat.

Systemgruppen

Systemgruppen sind spezielle, vordefinierte Gruppen. Diese Gruppen enthalten Daten über Kunden aus der Datenbank von Dialogue Server.

Hinzufügen einer neuen Systemgruppe

Klicken Sie im Domänenfenster auf der linken Seite auf die Gruppengliederung, und wählen Sie **Gruppe** > **Neue Systemgruppe...**, um eine neue Datengruppe zu einer Domäne hinzuzufügen. Daraufhin wird ein Fenster angezeigt, in dem aus den verfügbaren Systemgruppen ausgewählt werden kann. Es ist möglich, mehrere Gruppen gleichzeitig auszuwählen.

Sobald der Benutzer auf *OK* klickt, werden die Gruppe und ihre Felder der Domäne hinzugefügt. Die Eigenschaften von Systemgruppen und ihren Feldern sind nicht so flexibel wie bei normalen Datengruppen. Es ist jedoch möglich ihre Namen und Beschreibungen zu ändern. Die Webeigenschaften können ebenfalls geändert werden.

Die verschiedenen Systemgruppen

Derzeit sind fünf Systemgruppen im System verfügbar. Alle diese Gruppen sind 1:n-Gruppen. Zukünftig können noch weitere Gruppen hinzukommen.

Aktivitäten enthält Informationen über alle Aktivitäten, die mit einem Kunden verbunden sind.

Antwortformulare enthält Informationen über alle Antwortformulare, die mit einem Kunden verbunden sind. In dieser Gruppe gibt es eine Datenzeile pro Antwortformular.

Kategorien enthält Informationen über alle Kategorien, in denen ein Kunde Mitglied ist.

Dialoge enthält Informationen über Dialoge, an denen der Kunde teilnimmt. In dieser Gruppe gibt es eine Datenzeile pro aktivem Teilnehmer.

Nachrichten enthält eine Datenzeile pro an den Kunden gesendete Nachricht.

Felder

Ein Feld stellt einen einzigen Datenwert eines Kunden in der Domäne dar. Die wichtigsten Eigenschaften eines Feldes sind der Name und der Datentyp.

Die Datenquelle einer Gruppe definiert eine Reihe von Feldern, die in der Datengruppe enthalten sind. Dies sind normale Datenfelder. Außerdem kann eine Gruppe *Suchfelder* enthalten, bei denen es sich um virtuelle Felder handelt, die auf von der Suchquelle abgerufenen Daten basieren. Suchfelder werden in einem eigenen Abschnitt weiter unten beschrieben.

Das Fenster „Feldeigenschaften“

Doppelklicken Sie zum Bearbeiten der Eigenschaften eines Feldes auf das Feld, oder wählen Sie im Menü die Optionen **Feld** > **Eigenschaften...**. Das Fenster „Eigenschaften“ enthält Einstellungen zum Namen, Datentyp, Suchverhalten, logischem Inhalt und Web sowie zur Validierung:

--> **Allgemeine Eigenschaften**

--> **Sucheigenschaften**

--> **Validierungseigenschaften**

--> **Inhaltstypeigenschaften**

--> **Webeigenschaften**

Allgemeine Eigenschaften

Die erste Registerkarte zeigt Schlüsseleigenschaften an. *Feldname* ist der Name des Feldes, der in Anwendungen angezeigt wird. Er kann sich vom *Quellfeldnamen* unterscheiden, was der darunter liegende Feldname ist (meist der Name der zugehörigen Spalte in der Datenbank). Der Name kann aus Klein- und Großbuchstaben, Zahlen und dem Unterstrich bestehen. Der Name muss mit einem Buchstaben beginnen. Er darf **keine** Leerzeichen enthalten. Namen von **Ausdrucksfunktionen** sind reserviert und können nicht als Feldnamen verwendet werden. Außerdem muss der Name konform sein mit der Namenssyntax der **Extensible Markup Language (XML) 1.0 (Vierte Edition)**. Sie müssen die Funktion „Kundendomäne überprüfen“ verwenden, um den Namen zu validieren.

Nur für fortgeschrittene Anwendung sollte aktiviert sein, wenn das Feld normalerweise für den Benutzer nicht sichtbar ist.

Datentyp ist der Datentyp des Feldes. Siehe Abschnitt **Datentypen** für eine Liste der gültigen Typen.

Größe ist die physikalische Größe des Feldes. Dies wird normalerweise automatisch festgelegt, wenn die Feldliste aktualisiert wird.

Schreibgeschützt bestimmt, ob das Feld aktualisierbar ist. Diese Option ist nur aktiviert, wenn die Datengruppe aktualisierbar ist.

Sucheigenschaften

Sucheigenschaften

Die Registerkarte „Suchen“ erlaubt es dem Benutzer, eine Quelle anzugeben, von der aus nach Werten gesucht werden kann. Dies bedeutet, dass die Werte aus der Suchquelle abgerufen und dem Feld zugewiesen werden, wenn der Benutzer beispielsweise einen Kunden in Customer View bearbeitet.

Eine Suchquelle ist ein Datensatz aus Spalten und Zeilen mit Daten. Jede Zeile enthält einen möglichen Suchwert.

Das *Schlüsselfeld* ist ein weiteres Feld in der gleichen Datengruppe. Das *Suchschlüsselfeld* ist eine Spalte in der Suchquelle, in diesem Fall *PC_CODE*. Während einer Suche muss der Wert im *Schlüsselfeld* dem Wert im *Suchschlüsselfeld* entsprechen. Daher sollte das *Suchschlüsselfeld* eindeutige Werte enthalten, was bedeutet, dass ein Wert im *Schlüsselfeld einer oder keiner* Zeile entspricht.

Wenn eine Suche ausgeführt und ein Treffer gefunden wird, sollte der Wert des Feldes, in diesem Fall „Stadt“, dem Suchergebnisfeld entsprechen. Das *Suchergebnisfenster* ist eine Spalte im durchsuchten Quelldatensatz, die das Ergebnis der Suche angibt.

Zusammen definieren die Sucheinstellungen, dass Werte für dieses Feld gefunden werden können, indem der Wert des *Schlüsselfeldes* verwendet und in der Suchquellenach einer Übereinstimmung mit *PC_CODE* (Suchschlüsselfeld) gesucht wird.

Felder, die eine Suche benutzen versus Suchfelder

Es ist möglich, ein spezielles Feld, genannt *Suchfeld*, zu definieren, das nicht in der Datenquelle der Gruppe existiert, sondern ein so genanntes *virtuelles Feld ist.* Ein Suchfeld muss eine definierte Suche besitzen, was bedeutet, die Registerkarte „Suchen“ des Eigenschaftsfensters muss konfiguriert sein.

Im folgenden Beispiel wird angenommen, dass in der Hauptdatengruppe ein Feld *pers_pms_id* existiert, das eine Nummer enthält, die den Familienstand des Kunden darstellt. Anstatt dem Benutzer das Feld *pers_pms_id* anzuzeigen, wird ein Suchfeld erstellt, das *Familienstand* heißt. Wählen Sie zur Erstellung eines neuen Suchfeldes die Optionen **Feld** > **Neues Suchfeld...**. Im Voraus wurde eine Suchquelle definiert, welche den folgenden Datensatz enthält:

pms_id	pms_desc
1	Ledig
2	Verheiratet
3	Zusammenlebend

Um der Werte für *Familienstand* zu suchen, wird der Wert für *pers_pms_is* verwendet, um eine Übereinstimmung im Suchdatensatz zu finden.

In Customer View könnte das Suchfeld *Familienstand* anstelle von *pers_pms_id* angezeigt werden. Auf der Suchseite wird ein Dropdownfeld angezeigt, und der Benutzer kann zwischen *ledig, verheiratet und zusammenlebend* auswählen. Außerdem kann der Benutzer auf der Seite „Kundendaten bearbeiten“ aus den Werten für *Familienstand* auswählen. Wenn Aktualisierungen angewendet werden, aktualisiert die Anwendung *pers_pms_id* im Hintergrund mit der korrekten ID – unsichtbar für den Benutzer.

Suchfilterung (nur 1:n-Gruppen)

Die Option *Suchwerte filtern* ermöglicht die Filterung von Suchwerten basierend auf dem Wert eines anderen Feldes in der Datengruppe. Bei der Konfiguration der Suchfilterung wird es typischerweise von Benutzeranwendungen verwendet, um die Werte von Dropdown-Optionsfeldern zu filtern. Die Suchfilterung wird im Auswahldesigner in Visual Dialogue und in Customer View (Suchseite und 1:n-Bearbeitungsseite) unterstützt. Die Suchfilterung wird momentan nur für Felder in 1:n-Gruppen unterstützt. Bei der Konfiguration der Suchfilterung definiert das durch das *Filterfeld* verwiesene Feld den Filter. Nur Felder innerhalb einer 1:n-Gruppe können ausgewählt werden als das *Filterfeld.* Außerdem können nur Felder ausgewählt werden, die eine Suche besitzen. *Filterschlüsselfeld* ist das Feld in der *Suchquelle*, das für die Filterung der Suchwerte verwendet wird.

Validierungseigenschaften

Validierungseigenschaften

Aktualisierbare Felder können Regeln für die Validierung haben. Diese werden auf der Registerkarte „Validierung“ eingerichtet.

Standardwert zuweisen kann aktiviert werden, wenn ein Standardwert geeignet ist. In diesem Fall wird Customer View automatisch den *Standardwert* übernehmen, wenn neue Reihen eingefügt werden.

Erforderlich muss aktiviert sein, wenn das Feld nicht leer sein darf.

Min. Wert/Länge und *Max. Wert/Länge* werden bei numerischen und Zeichenfolgefeldern verwendet. Diese definieren die minimalen und maximalen Werte bei numerischen Feldern und die minimale und maximale Länge bei Zeichenfolgefeldern.

Ein regulärer Ausdruck kann ebenfalls zur Validierung von Werten verwendet werden. Ein *reguläres Ausdrucksmuster* wird beispielsweise definiert, um eine E-Mail-Adresse zu validieren. *Fehlermeldung für regulären Ausdruck* enthält die Nachricht, die dem Benutzer angezeigt wird, falls die Validierung fehlschlägt. Es kann auf die Schaltfläche rechts vom regulären Ausdrucksmuster geklickt werden, um den Ausdruck zu testen.

Obwohl Dialogue Server die Feldvalidierung ausführt, implementiert Customer View die Validierungsregeln als JavaScript auf den Webseiten, um eine schnelle und benutzerfreundliche Schnittstelle zu erstellen.

Inhaltstypeigenschaften

Inhaltstypeigenschaften

Die Registerkarte „Inhaltstyp“ wird verwendet, um anzugeben, das ein Feld eine spezielle Bedeutung hat. Es gibt zwei Optionen. Erstens kann ein Feld so definiert werden, dass es auf eine andere Kundendomäne verweist. Das Feld *Kontakt-ID* in der Domäne *Unternehmen* kann beispielsweise so eingestellt werden, das es auf die Domäne *Kontakt* verweist. Dies bedeutet, dass Werte von *Kontakt-ID* den Werten des identifizierenden Feldes (Kunden-ID) in der Domäne *Kontakt* entsprechen müssen.

Die Option *Aktualisierungen der Domäne in Customer View aktivieren* steuert, ob Customer View in der Domäne *Unternehmen* Verknüpfungen (Schaltflächen) zum Einfügen, Bearbeiten und Löschen von Kunden in der verknüpften Domäne anzeigt. Diese Option gilt nur für 1:n-Gruppen.

Zweitens gibt es einen Satz von *sonstigen Feldinhaltstypen*. Diese werden für Felder wie E-Mail-Adressen, URLs und Telefonnummern verwendet, wenn die Werte Inhalte darstellen, die mit einer Anwendung geöffnet werden können, um spezifische Aktionen auszuführen. Wenn zum Beispiel *URL (Internetadresse)* ausgewählt ist, öffnet Customer View die Verknüpfung in einem neuen Browserfenster, wenn der Benutzer neben dem Feld auf ein kleines Symbol klickt.

Webeigenschaften

Interneteigenschaften

Die Registerkarte „Interneteigenschaften“ enthält Einstellungen, die für Internetanwendungen gelten.

Customer View

In Customer View anzeigen gibt an, ob das Feld in Customer View sichtbar ist. *In Customer View suchbar* gibt an, ob das Feld in Customer View als Suchkriterium angezeigt wird.

Kundenprofil

In Kundenprofil anzeigen gibt ab, ob das Feld im Webprofil des Kunden (verfügbar über Customer Web Access) angezeigt wird. Aktivieren Sie *Aktualisierung aus dem Kundenprofil zulassen*, wenn es dem Kunden gestattet sein soll, das Feld über seine Profilseite zu aktualisieren.

Telemarketing

Im Telemarketing-Web anzeigen bestimmt, ob das Feld in der Webanwendung „Telemarketing“ angezeigt wird. Aktivieren Sie *Aktualisierung aus Telemarketing-Web zulassen*, wenn es Telemarketing-Betreibern gestattet sein soll, das Feld zu aktualisieren.

Sucheinstellungen

Datensatz für Suchquelle zwischenspeichern gilt für Felder, die eine Suche verwenden, und bestimmt, ob der Suchdatensatz von den Webanwendungen zwischengespeichert wird. Dies ist geeignet, wenn der Suchdatensatz nicht zu groß ist und sich normalerweise nicht ändert.

Als Dropdownmenü anzeigen bestimmt, ob das Feld mit einem Dropdownmenü aller Werte des Suchdatensatzes angezeigt wird.

Datentypen

Einleitung

In Dialogue Admin ist eine Reihe von Datentypen definiert. Diese Typen werden bei der Definition von Kundendomänen als logische Datentypen verwendet.

Datentypen

Die Tabelle unten beschreibt die unterschiedlichen Datentypen.

Datentypname	Beschreibung
Zeichenfolge	Zeichenfolge-Datentyp
ganze Zahl	Ganzzahl-Datentyp
Fließkomma	Gleitkomma-Datentyp
Datum/Uhrzeit	Datum/Zeit-Datentyp
date	Datum-Datentyp (enthält nicht die Zeitinformation aus „datetime“)
boolean	boolescher Datentyp
binär	Binär-Datentyp
Int64	Ganzzahliger 64-Bit-Datentyp zwischen Ganzzahl und Gleitkomma

Suchquellen

Informationen über Suchquellen

Eine Suchquelle ist eine Datenquelle, die verwendet wird, wenn Suchen in Datenfeldern definiert werden. In der aktuellen Version basieren alle Suchquellen auf SQL-Abfragen, die im SQL-Repository gespeichert sind.

Informationen über die SQL-Suche

Eine SQL-Suchabfrage ist eine SELECT-Anweisung, die einen Satz von Spalten zurückgibt, einschließlich *Suchschlüsselfeld* und *Suchergebnisfeld*. Eine Suchquelle kann von mehreren Feldern verwendet werden. Daher enthält sie oft mehr als zwei Spalten.

Definieren einer neuen Suchquelle

Folgende Schritte sind erforderlich, um eine neue Suchquelle zu definieren.

1. Vergewissern Sie sich, dass die SQL-Suche im SQL-Repository stattfindet.
2. Öffnen Sie das Fenster *Suchquellen definieren*, indem Sie **Datei** > **Suchquellen definieren** auswählen.
3. Klicken Sie anschließend auf die Schaltfläche *Neu...*, und benennen Sie die neue Suchquelle.
4. Wählen Sie die zu verwendende SQL.
5. Klicken Sie auf *Felder aktualisieren*, um eine Liste mit Spalten aus der SQL-Anweisung abzurufen.
6. Klicken Sie auf *Schließen*, um zum Hauptfenster der Kundendomäne zurückzukehren.

Die Suchquelle und ihre Felder sind nun verfügbar, wenn Suchen im Fenster **Feldeigenschaften** eingerichtet werden.

Kundendomänenüberprüfung

Das Konfigurieren von Kundendomänen kann eine komplexe Aufgabe sein, besonders, wenn Sie mit großen Domänen arbeiten. Sie haben große Freiheit bei der Konfiguration einer Kundendomäne, um auf verschiedene Quellen von Kundendaten zuzugreifen. Diese Freiheit kann jedoch dazu führen, dass Konfigurationen erstellt werden, die nicht richtig oder gar nicht funktionieren. Sie können das Tool *Kundendomänenüberprüfung* verwenden, um einige dieser Probleme zu vermeiden.

Weitere Informationen finden Sie in den folgenden Themen:

--> **Ausführen der Kundendomänenüberprüfung**

--> **Kundendomänenüberprüfung – Nachrichten**

Ausführen der Kundendomänenüberprüfung

Die Kundendomänenüberprüfung wird aus dem Fenster **Kundendomäne bearbeiten** ausgeführt. Klicken Sie zum Starten der Überprüfung auf die Schaltfläche *Überprüfen* oder drücken Sie *F9*. Nach der Überprüfung kann das Ergebnis in einem Bereich unter der Felderliste angezeigt werden. Dieser Bereich wird eine Liste von Nachrichten anzeigen, die (mögliche) Probleme in der Domänenkonfiguration beschreiben. Sie können auf eine Überprüfungsnachricht in der Liste doppelklicken oder *Eingabe* drücken, um diesen Problembereich in der Domänenkonfiguration zu fokussieren.

Kundendomänenüberprüfung – Nachrichten

Einleitung

Wenn Sie die Kundendomänenüberprüfung ausführen, wird eine Liste von Nachrichten angezeigt, welche die möglichen Probleme mit der Domänenkonfiguration anzeigen. Die untere Tabelle beschreibt diese Nachrichten detailliert.

Typ	Nachricht	Beschreibung
Warnung	Für das Feld <*Feldname*> muss ein Standardwert angegeben werden.	Sie haben ein Datenfeld so konfiguriert, dass es einen Standardwert verwendet, aber der angegebene Wert ist leer.
Warnung	Für Parameter <*Parametername*> muss ein Standardwert angegeben werden.	Sie haben eine Parameterbindung so konfiguriert, dass ein konstanter Parameterwert verwendet wird, aber der angegebene Wert ist leer.
Tipp	Ein Domänenname sollte mit einem alphanumerischen Zeichen beginnen.	Dies ist kein Fehler. Aber es ist eine übliche Vorgehensweise, einen Domänennamen entweder mit einem Buchstaben oder einer Zahl zu beginnen.
Fehler	Eine Domänen-Webadressengruppe darf keine Gruppe des booleschen Typs sein.	Wenn Sie eine Domäne so konfigurieren, dass sie eine Webadressengruppe besitzt, darf diese Gruppe nicht vom booleschen Typ sein.
Fehler	Eine Domänen-Webadressengruppe des Typs „1:n“ muss den Bearbeitungsmodus im Tabulatorformat verwenden.	Wenn Sie die Domäne so konfigurieren, dass Sie eine Webadressengruppe besitzt, muss diese Datengruppe so eingerichtet sein, dass sie den Webbearbeitungsmodus im Registerformat verwendet.
Fehler	Eine Hauptdomänengruppe muss zur Aktualisierung von Daten ein Plug-In verwenden, wenn die Prüfung auf Duplikate aktiviert ist.	Sie haben die Prüfung auf Duplikate in der Domäne aktiviert. Ihre Hauptdatengruppe ist aber nicht so eingerichtet, dass ein Plug-In zur Datenaktualisierung verwendet wird.
Fehler	Für die Domäne muss ein Name angegeben werden.	Sie haben keinen Namen für die Domäne angegeben.
Fehler	Für das Feld muss ein Name angegeben werden.	Sie haben ein Datenfeld erstellt, aber keinen Namen dafür angegeben.
Fehler	Für die Gruppe muss eine Name angegeben werden.	Sie haben eine Datengruppe erstellt, aber keinen Namen angegeben.
Warnung	In der verknüpften Definition der Kundendomäne für <*Feldname*> hat das identifizierende Feld einen anderen Datentyp als das Feld selbst.	Sie haben ein Datenfeld so konfiguriert, dass es auf eine andere Kundendomäne verweist. Der Datentyp des Feldes ist aber anders als der Datentyp des identifizierenden Feldes in der Domäne, auf die Sie verweisen. Dies kann zu geringer Leistung und Fehlern führen.

Warnung	In der Definition der Suche für <*Feldname*> sind die Datentypen des Suchschlüsselfeldes und des Schlüsselfeldes unterschiedlich.	Sie haben eine Datenquelle so konfiguriert, dass sie eine Datenquelle verwendet. Der Datentyp des Suchschlüsselfeldes ist aber anders als der Datentyp des Schlüsselfeldes. Dies kann zu geringer Leistung und Fehlern führen.
Warnung	In der verknüpften Definition der Kundendomäne für <*Feldname*> hat das identifizierende Feld einen anderen Datentyp als das Feld selbst.	Sie haben ein Datenfeld so konfiguriert, dass es eine Suchquelle verwendet. Der Datentyp des Suchergebnisses ist aber nicht derselbe, wie der Datentyp des Feldes selbst. Dies kann zu geringer Leistung und Fehlern führen.
Warnung	In der Parameterdefinition für <*Parametername*> sind die Datentypen des gebundenen Feldes und des Parameterfeldes unterschiedlich.	Sie haben eine Parameterbindung so konfiguriert, dass sie an ein Feld bindet. Der Datentyp des Parameterfeldes und des Feldes, an das es gebunden ist, sind aber unterschiedlich. Dies kann zu geringer Leistung und Fehlern führen.
Fehler	Das gebundene Feld des Parameters <*Parametername*> wurde nicht in der übergeordneten Datengruppe gefunden.	Sie haben eine Parameterbindung so konfiguriert, dass sie an ein Feld bindet. Das Feld wird aber in der übergeordneten Gruppe nicht gefunden. Dies kann passieren, wenn das Feld umbenannt oder aus der übergeordneten Datengruppe gelöscht wurde, nachdem die Parameterbindung definiert wurde.
Warnung	Der Standardwert kann nicht in den Felddatentyp des Feldes <*Feldname*> umgewandelt werden.	Sie haben ein Datenfeld so konfiguriert, dass ein Standardwert verwendet wird. Der angegebene Wert kann aber nicht in den Datentyp des Feldes umgewandelt werden.
Warnung	Der Standardwert konnte nicht in den Felddatentyp des Parameters <*Feldname*> umgewandelt werden.	Sie haben eine Parameterbindung so konfiguriert, dass ein Standardwert verwendet wird. Der angegebene Standardwert kann aber nicht in den Datentyp des Feldes umgewandelt werden.
Tipp	Die Domäne ist nicht aktiviert.	Sie haben Ihre Domäne nicht aktiviert. Dies bedeutet, dass auf die Domäne nicht aus Visual Dialogue oder Web Applications zugegriffen werden kann. Beachten Sie, dass die Domäne standardmäßig deaktiviert ist, wenn Sie sie erstellen.
Fehler	Die Domäne muss mindestens eine Datengruppe haben.	Sie haben keine Datengruppen für die Domäne definiert. Die Domäne muss mindestens eine Datengruppe haben, um Kundendaten abrufen zu können.
Fehler	Der Domänenname enthält einen unzulässigen Wert.	Der Name der Kundendomäne darf keine Punkte, Anführungszeichen oder doppelte Anführungszeichen enthalten.
Fehler	Die Domänen-Webadressengruppe <*Gruppenname*> wurde nicht in der Domäne gefunden.	Sie haben die Domäne so konfiguriert, dass eine Webadressengruppe verwendet wird. Die Datengruppe wurde aber nicht in der Domänendefinition gefunden. Dies kann

		passieren, wenn die Datengruppe umbenannt oder gelöscht wurde, nachdem Sie die Domäne für eine Webadressgruppe konfiguriert haben.
Fehler	Die Domänen-Webanzeigefeld <*Feldname*> wurde nicht in der Domäne gefunden.	Sie haben Ihre Domäne so konfiguriert, dass ein Webanzeigefeld verwendet wird. Das Datenfeld wurde aber nicht in der Domäne gefunden. Dies kann passieren, wenn das Datenfeld umbenannt oder gelöscht wurde, nachdem Sie die Domäne für die Verwendung eines Webanzeigefelds konfiguriert haben.
Fehler	Das Domänen-Webanzeigefeld darf nicht vom Typ „boolesch“ oder „binär“ sein.	Wenn Sie die Domäne so konfigurieren, dass sie ein Webanzeigefeld besitzt, darf der Datentyp dieses Feldes nicht „boolesch“ oder „binär“ sein.
Fehler	Das Webhinweisfeld <*Feldname*> der Domäne wurde nicht in der Domäne gefunden.	Sie haben Ihre Domäne so konfiguriert, dass ein Webhinweisfeld verwendet wird. Das Datenfeld wurde aber nicht in der Domäne gefunden. Dies kann passieren, wenn das Datenfeld umbenannt oder gelöscht wurde, nachdem Sie die Domäne für die Verwendung eines Webhinweisfelds konfiguriert haben.
Fehler	Das Webhinweisfeld der Domäne muss vom Typ „Zeichenfolge“ sein.	Wenn Sie die Domäne so konfigurieren, dass ein Webhinweisfeld verwendet wird, muss der Datentyp dieses Feldes „Zeichenfolge“ sein.
Fehler	Das Feld <*Feldname*> kann keine/n maximalen Wert/Länge haben.	Wenn Sie ein Datenfeld so konfigurieren, dass es eine/n maximale/n Wert/Länge hat, muss der Datentyp des Feldes „Zeichenfolge“, „Ganzzahl“, „int64“ oder „Gleitkommawert“ sein.
Fehler	Das Feld <*Feldname*> kann keine/n minimalen Wert/Länge haben.	Wenn Sie ein Datenfeld so konfigurieren, dass es eine/n minimale/n Wert/Länge hat, muss der Datentyp des Feldes „Zeichenfolge“, „Ganzzahl, „int64“ oder „Gleitkommawert“ sein.
Fehler	Das Feld <*Feldname*> soll laut Einstellung anhand eines regulären Ausdrucks validieren; es wurde jedoch kein regulärer Ausdruck festgelegt.	Sie haben ein Datenfeld so konfiguriert, dass es anhand eines regulären Ausdrucks validiert. Das angegebene, reguläre Ausdrucksmuster ist aber leer.
Fehler	Der Feldname <*Feldname*> enthält eines oder mehrere unzulässige Zeichen.	Der Name eines Datenfeldes darf nur die Buchstaben A bis Z, '', Zahlen und Unterstriche enthalten.
Warnung	Der Feldname <*Feldname*> ist ein Ausdrucksfunktionsname.	Der Name eines Datenfeldes darf nicht der einer Ausdrucksfunktion sein. Dies kann zu Fehlern oder ungewollten Ergebnissen im internen Ausdrucksanalysierer führen.

Fehler	Die Gruppe <*Gruppenname*> besitzt keine Datenfelder.	Sie haben keine Datenfelder für die Datengruppe definiert. Wählen Sie aus dem Menü *Feldliste aktualisieren*, um der Datengruppe Felder hinzuzufügen.
Fehler	Die Gruppe <*Gruppenname*> hat keine Datenquelle.	Sie haben keine Datenquelle für die Datengruppe ausgewählt. Die Datengruppe muss eine Datenquelle besitzen, um Kundendaten abrufen zu können.
Warnung	Die Gruppe <*Gruppenname*> hat keine Parameterbindungen.	Sie haben keine Parameterbindungen für die Datengruppe eingerichtet. Alle Datengruppen außer die Hauptgruppe sollten Parameterbindungen besitzen, um anzugeben, wie Daten in dieser Gruppe mit den Daten in der übergeordneten Datengruppe zusammenhängen.
Warnung	Die Gruppe <*Gruppenname*> ist aktualisierbar, alle Felder in der Gruppe sind jedoch schreibgeschützt.	Sie haben ein Datenfeld so konfiguriert, dass es aktualisierbar ist. Alle Datenfelder in der Gruppe sind aber schreibgeschützt. Dies bedeutet, dass Sie keine Daten in der Gruppe bearbeiten können.
Fehler	Der Feldname <*Feldname*> enthält eines oder mehrere unzulässige Zeichen.	Der Name eines Datenfeldes darf nur die Buchstaben A bis Z, '', Zahlen und Unterstriche enthalten.
Warnung	Der Gruppenname <*Gruppenname*> ist ein Ausdrucksfunktionsname.	Der Name einer Datengruppe darf nicht der einer Ausdrucksfunktion sein. Dies kann zu Fehlern oder ungewollten Ergebnissen im internen Ausdrucksanalysierer führen.
Fehler	Das für Suchfeld <*Feldname*> definierte Schlüsselfeld <*Feldname*> wurde nicht in der Datengruppe gefunden.	Sie haben eine Suche für ein Datenfeld konfiguriert. Das Schlüsselfeld wurde aber nicht in der Datengruppe gefunden, zu der das Feld gehört. Dies kann passieren, wenn das Schlüsselfeld umbenannt oder aus der Datengruppe gelöscht wurde, nachdem das Suchfeld gelöscht wurde.
Fehler	Die verknüpfte Kundendomäne, die in Feld <*Feldname*> verwendet wurde, ist nicht gültig: <*Fehlermeldung*>.	Sie haben ein Datenfeld so konfiguriert, dass es auf eine andere Kundendomäne verweist. Auf die verknüpfte Kundendomäne kann aber nicht zugegriffen werden. In der Fehlermeldung finden Sie weitere Informationen zur verknüpften Domäne.
Fehler	Das für Suchfeld <*Feldname*> definierte Schlüsselfeld <*Feldname*> wurde nicht in der Datengruppe gefunden.	Sie haben eine Suche für ein Datenfeld konfiguriert. Das Suchschlüsselfeld wurde aber nicht in der Suchquelle gefunden. Dies kann passieren, wenn das Suchschlüsselfeld umbenannt oder aus der Suchquelle gelöscht wurde, nachdem die Suche definiert wurde.
Fehler	Das für die Suche des Feldes <*Feldname*> definierte Suchergebnisfeld <*Feldname*> wurde nicht in der Suchquelle gefunden.	Sie haben für ein Datenfeld eine Suche konfiguriert. Das Suchergebnisfeld wurde aber nicht in der Suchquelle gefunden. Dies kann passieren, wenn das Suchergebnisfeld umbenannt oder aus der Suchquelle gelöscht wurde, nachdem die Suche definiert wurde.

Fehler	Die für die Suche des Feldes *<Feldname>* definierte Suchquelle *<Suchquelle>* wurde nicht in der Domäne gefunden.	Sie haben eine Suche für ein Datenfeld konfiguriert. Die Suchquelle wurde aber nicht in der Domäne gefunden. Dies kann passieren, wenn die Suchquelle umbenannt oder gelöscht wurde, nachdem die Suche definiert wurde.
Fehler	Das für die Suche des Feldes *<Feldname2>* definierte Suchfilterschlüsselfeld *<Feldname1>* wurde nicht in der Suchquelle gefunden.	Sie haben eine Suche mit Suchfilterung für ein Feld konfiguriert. Das Suchfilterschlüsselfeld wurde aber nicht in der Suchquelle gefunden. Dies kann passieren, wenn das Feld umbenannt oder aus der Suchquelle gelöscht wurde, nachdem die Suche definiert wurde.
Fehler	Feld '*<Feldname>*' ist mit einem Suchfilter konfiguriert. Dies ist nur zulässig, wenn das Feld Teil einer 1:n-Gruppe ist.	Die Suchfilterung wird nur für 1:n-Gruppen unterstützt. Sie haben möglicherweise die Suchfilterung für eine 1:n-Gruppe konfiguriert, und diese Gruppe wurde danach in eine 1:1-Gruppe geändert.
Fehler	'Das für die Suche von Feld *<Feldname2>* definierte Suchfilterfeld *<Feldname1>* wurde nicht in der Datengruppe gefunden.	Sie haben eine Suche mit Suchfilterung für ein Feld konfiguriert. Das Filterfeld wurde aber nicht in der Datengruppe gefunden, zu der das Feld gehört. Dies kann passieren, wenn das Filterfeld umbenannt oder aus der Datengruppe gelöscht wurde, nachdem die Suche definiert wurde.
Fehler	'Das für die Suche von Feld '*<Feldname2>*' definierte Suchfilterfeld '*<Feldname1>*' besitzt keine Suchquelle.	Sie haben eine Suche mit Suchfilterung für ein Feld konfiguriert, aber das Filterfeld besitzt keine Suche (die erforderlich ist). Dies kann passieren, wenn die Suche des Filterfeldes nach der Definition der Suche entfernt wurde.
Warnung	Der Datentyp des identifizierenden Feldes der Hauptdomänengruppe unterscheidet sich vom Datentyp „mh_customer_id“.	Das identifizierende Datenfeld in der Hauptdatengruppe wird verwendet, um Kundendaten in der Domäne mit anderen Kundendaten in Dialogue Database zu verknüpfen. Wenn sich der Datentyp des identifizierenden Feldes vom Datentyp des Kundenidentifizierers irgendwo anders im System unterscheidet, können geringe Leistung und Fehler auftreten.
Fehler	Die Hauptdomänengruppe muss ein identifizierendes Feld besitzen.	Kundendaten in der Domäne sind mit anderen Kundedaten in Dialogue Database durch das identifizierende Feld in der Hauptdatengruppe verbunden. Dies bedeutet, dass Sie ein identifizierendes Feld für die Hauptdomänengruppe festlegen müssen, damit die Domäne funktioniert.
Fehler	Die maximale Länge für das Feld *<Feldname>* ist ungültig.	Sie haben eine maximale Länge für das Datenfeld konfiguriert. Der Wert ist aber ungültig. Maximale Längen können nur für Felder des Typs „Zeichenfolge“ festgelegt werden und müssen ein ganzzahliger Wert sein.
Fehler	Die maximale Wert für das Feld *<Feldname>* ist ungültig.	Sie haben einen Maximalwert für das Datenfeld konfiguriert. Der Wert ist aber ungültig. Maximalwerte können nur für Felder des Typs „Ganzzahl“, „int64“ oder „Gleit-

		kommawert“ festgelegt werden, und der Wert ist ungültig, wenn er nicht in den Datentyp des Feldes umgewandelt werden kann.
Fehler	Die minimale Länge für das Feld <*Feldname*> ist ungültig.	Sie haben eine minimale Länge für das Datenfeld konfiguriert. Der Wert ist aber ungültig. Minimale Längen können aber nur für Felder des Typs „Zeichenfolge“ festgelegt werden und müssen ganzzahlig sein.
Fehler	Der minimale Wert für das Feld <*Feldname*> ist ungültig.	Sie haben einen minimalen Wert für das Datenfeld konfiguriert. Der Wert ist aber ungültig. Minimalwerte können nur für Felder des Typs „Ganzzahl“, „int64“ oder „Gleitkommawert“ festgelegt werden, und der Wert ist ungültig, wenn er nicht in den Datentyp des Feldes umgewandelt werden kann.
Fehler	Das Parameterfeld des Parameters <*Parametername*> wurde nicht in der Datengruppe gefunden.	Sie haben eine Parameterbindung konfiguriert. Das Parameterfeld wurde aber nicht in der Datengruppe gefunden. Dies kann passieren, wenn das Feld umbenannt oder aus der Gruppe gelöscht wurde, nachdem die Parameterbindung definiert wurde.
Fehler	Die Validierung des regulären Ausdrucks in Feld <*Feldname*> wird nicht kompiliert: <*Fehlermeldung*>.	Sie haben ein Datenfeld so konfiguriert, dass es einen regulären Ausdruck validiert. Es ist aber ein Fehler im regulären Ausdruck vorhanden, und er wird nicht kompiliert. In der Fehlermeldung finden Sie weitere Informationen zum regulären Ausdruck.
Fehler	Es gibt zwei oder mehr Felder mit dem Namen <*Feldname*>.	Zwei oder mehr Felder haben in der Datengruppe den gleichen Namen. Jeder Feldname in einer Datengruppe muss eindeutig sein.
Fehler	Es gibt zwei oder mehr Gruppen mit dem Namen <*Gruppenname*>.	Sie haben zwei oder mehr Datengruppen mit dem gleichen Gruppennamen. Jeder Gruppenname in einer Domäne muss eindeutig sein.

Formeln

Einführung in Ausdrücke

Ausdrücke werden zum Zugriff auf Daten in Kundendomänen verwendet.

Die *Ausdruckssprache* besitzt eine formale Syntax-Definition, die für leichtes Verständnis und Benutzerfreundlichkeit entwickelt wurde.

Ausdrücke werden in verschiedenen Teilen der Produktsuite verwendet.

Ausdrücke und Auswahlen

Ausdrücke werden in Auswahlen zur Definition von Kriterien verwendet. Im Auswahldesigner in Visual Dialogue definiert genau ein Ausdruck eine Auswahl. Der Benutzer erstellt diesen Ausdruck entweder durch das Hinzufügen von Kriterien in der Designansicht oder durch das Bearbeiten des Ausdrucks direkt in der Ausdrucksansicht.

Das folgende Beispiel dient der Auswahl aller Kunden mit E-Mail-Adresse:

has EmailAddress

Ausdrücke und Webanwendungen

Sie können Ausdrücke in den Webanwendungen verwenden, um erweiterte Suchkriterien zu definieren.

Ausdrücke und Nachrichten

Wenn Sie mit Nachrichtenvorlagen und Basisnachrichten arbeiten, können Sie Ausdrücke innerhalb eines Vorlagentextes verwenden. So werden beispielsweise Ausdrucksfunktionen in einer E-Mail zur Generierung einer Antwortformular-URL verwendet:

```
«#AnswerFormURL(1048, 1101, -1, -1, TRUE)»
```

Hinweis: Das Zeichen „#“ signalisiert dem System, das Mergefeld als einen Ausdruck und nicht als ein einfaches Datenfeld wie „FirstName“ zu behandeln.

Ausdrücke und Dialogue Admin

Dialogue Admin bietet das Werkzeug *Ausdrucksanalyse.* Dieses Werkzeug hilft Ihnen, Ausdrücke zu testen.

Definitionen der Ausdruckssyntax

Hauptgruppenausdrücke

Die *Hauptgruppe* ist die oberste Gruppe in der Definition der *Kundendomäne*. Typischerweise ist dies die Personengruppe, die Daten wie Vorname, Nachname usw. zurückgibt. Sie können direkt die Feldnamen verwenden, um Ausdrücke zu schreiben, die auf Daten der Hauptgruppe zugreifen.

Beispiele:

- Firstname = "Max"
- Firstname = "Max" And Lastname = "Mustermann"

Sie können auch ein Präfix bei Hauptgruppennamen verwenden:

- Person.Firstname = "Max"

1:1-Gruppenausdrücke

1:1-Gruppen sind Datengruppen, die nur eine oder keine Datenzeile pro Kunden zurückgeben. Typischerweise ist dies die Hauptpostadresse oder der Arbeitgeber, der mit einer Person verbunden ist. Diese Gruppen werden in der Domäne so definiert, dass sie nur eine Zeile zurückgeben.

Für den Zugriff auf diese Daten verwenden Sie den Feldnamen, der mit dem Gruppennamen als Präfix kombiniert wurde.

Beispiele:

- Address.Streetname = "Musterstraße"
- Company.Name = "Mustermann GmbH"

1:n-Gruppenausdrücke

Datengruppen, die mehrere Zeilen pro Kunde zurückgeben, heißen *1:n-Gruppen*. Solche Gruppen von Kundendaten sind typischerweise Aktivitäten, Dokumente oder Antwortformulare. Diese Gruppen werden in der Domäne so definiert, dass sie mehrere Zeilen zurückgeben (siehe auch Abschnitt „Gruppeneigenschaften“).

Sie müssen auf vorhandene Daten prüfen, um Ausdrücke zu verwenden, die auf 1:n-Gruppen zugreifen. Der Ausdrucksoperator **has** ist essentiell. Dieser Operator überprüft, ob eine Datengruppe Informationen enthält, und gibt *true* oder *false* zurück.

Beispiele:

- has Documents
- Has Documents[DocName = "Bericht*"]

Das erste Beispiel oben gibt alle Kunden zurück, die *mindestens* ein Dokument haben. Das zweite Beispiel gibt alle Kunden zurück, die ein Dokument haben, dessen Name mit „Bericht“ beginnt.

Andere Beispiele:

- has Documents[DocId = 1000 and DocType = "Brief"]
- Has Documents[DocDate > '01.01.2002']
- Numberof(Activities) > 2

Beachten Sie: Das letzte Beispiel verwendet eine *Ausdrucksfunktion*, genannt **NumberOf(...)**, welche die Anzahl der Elemente in der 1:n-Gruppe *Aktivitäten* zurückgibt.

Zahlenwerte und das Dezimaltrennzeichen in Ausdrücken

Das Zeichen „.“ (amerikanischer Standard) dient in Ausdrücken als Dezimaltrennzeichen. Das Zeichen „ , “ (Komma) dient zum Trennen von Funktionsparametern.

- CustomerScore > 2.5

Datums- und Zeitwerte in Ausdrücken

Konstante Datums- und Zeitwerte werden in einfachen Anführungszeichen eingeschlossen. Die formale Syntax von Datums- und Zeitwerten richtet sich nach der XML-Syntax:

- 'YYYY-MM-DDTHH:MM:SS', Beispiel: '2003-01-28T17:00:00'

Der Zeitteil kann natürlich entfernt werden:

- 'YYYY-MM-DD', Beispiel: '2003-01-28'

Zusätzlich wird das aktuelle, regionale Datums- bzw. Zeitformat von Dialogue Server unterstützt. Beispiele (deutsches Datums- und Zeitformat):

- '28.12.2003 17:00'
- '28.12.2003'

Ausdrucksoperatoren

Operatoren und Ausdrücke

Ausdrücke verfügen über eine Reihe von integrierten Operatoren, die als Basiselemente verwendet werden. Es stehen verschiedene Arten von Operatoren zur Verfügung.

Vergleichsoperatoren

Die folgenden Operatoren werden verwendet, um Werte in Ausdrücken zu vergleichen.

=
Prüft, ob zwei Werte *gleich* sind.
>
Prüft, ob der linksseitige Wert *größer als* der rechtsseitige Wert ist.
<
Prüft, ob der linksseitige Wert *kleiner* als der rechtsseitige Wert ist.
>=
Prüft, ob der linksseitige Wert *größer als oder gleich* dem rechtsseitigen Wert ist.
<=
Prüft, ob der linksseitige Wert *kleiner als oder gleich* dem rechtsseitigen Wert ist.
<>
Unterschied zu prüft, ob zwei Werte *unterschiedlich* sind.

Logische Operatoren

Die folgenden Operatoren sind logische Operatoren, die in der booleschen Algebra verwendet werden:

NICHT
Der logische Operator „NOT“ gibt „false“ zurück, wenn der darauf folgende Wert wahr ist und andersherum.
AND
Gibt „true“ zurück, wenn sowohl der Wert auf der rechten als auch auf der linken Seite wahr ist.

Oder

Gibt „true“ zurück, wenn entweder der Wert auf der rechten oder auf der linken Seite wahr ist.

Arithmetische Operatoren

Die folgenden Operatoren sind arithmetische Operatoren, die zur Manipulation von Zahlenwerten verwendet werden:

+

Gibt die Summe zweier Werte zurück.

-

Gibt die Differenz zweier Werte zurück.

*

Gibt das Produkt zweier Werte zurück.

/

Gibt den Quotienten zweier Werte zurück.

Zeichenfolgenopertor „+“

Der Operator „+“ kann neben dem Addieren von Zahlen auch zum Verketten von Zeichenfolgen verwendet werden. Beispiel:

- Nachname + ', '+ Vorname

Dieser Ausdruck gibt Werte zurück wie: „Mustermann, Max“

Operator „has“

Der Operator „has“ ist in Ausdrücken wichtig und hängt vom darauf folgenden Argument ab. Er gibt immer *true* oder *false* zurück.

Wenn das Argument ein Datenfeld ist und einen Wert hat, ist der zurückgegebene Wert „true“. Beispiele:

- **has** EmailAddress
- **has** Address.Postcode

Wenn das Argument eine Datengruppe ist und diese Datengruppe für den gegebenen Kunden mindestens ein Element (oder eine Zeile) enthält, ist der zurückgegebene Wert „true“. Beispiele:

- **has** Address
- **has** Documents
- **has** Activities[ChannelType = "SMS"]

Benennungsoperator „:“

Der Operator „:“ wird verwendet, um zu benennen oder individuelle Kriterien zu beschreiben. Beispiel:

- "Kunde mit E-Mail-Adresse": has email
- "möchte mehr Informationen: Ja": BoolAnswer(1000, 10, 1)

Die Hauptanwendung dieses Operators liegt im Auswahldesigner, um Kriterien eine benutzerfreundliche Beschreibung zu geben.

Ausdrucksdatentypen

Einleitung

Ein Satz von Datentypen, der von Ausdrücken und Ausdrucks-Plug-Ins verwendet wird.

Datentypen

Die Tabelle unten beschreibt die unterschiedlichen Datentypen.

Datentypname	Beschreibung
Zeichenfolge	Zeichenfolge-Datentyp
ganze Zahl	Ganzzahl-Datentyp
Fließkomma	Gleitkomma-Datentyp
Datum/Uhrzeit	Datum/Zeit-Datentyp
date	Datum-Datentyp (enthält nicht die Zeitinformation aus „datetime“)
boolean	boolescher Datentyp

Ausdrucksfunktionen

Funktionen und Ausdrücke

Ausdrücke enthalten eine große Anzahl von Funktionen. Einige Funktionen werden verwendet, um Kundendomänendaten zu ändern. Andere sind beispielsweise mit Fragebögen und Customer Web Access verbunden.

Gruppen von Ausdrucksfunktionen

Folgende Kategorien von Ausdrucksfunktionen sind verfügbar:

- **Aggregatfunktionen**

  Diese Kategorie von Ausdrucksfunktionen unterstützt die Aggregation von Daten.

- **Analytics-Funktionen**

  Diese Kategorie von Ausdrucksfunktionen unterstützt die Integration mit Portrait Customer Analytics.

- **Inhaltsobjektfunktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um auf Inhaltsobjekte zuzugreifen und diese auszuführen.

- **Customer Web Access-Funktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um Kunden beim Zugriff auf Webanwendungen zu unterstützen. Einige der Funktionen erzeugen URLs zu personalisierten Webseiten von Dialogue Server. Diese Funktionen werden häufig in Nachrichtenvorlagen verwendet.

- **Datums- und Zeitfunktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um mit Datums- und Zeitwerten zu arbeiten.

- **Dialogfunktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um mit Dialogen und Dialogteilnehmern zu arbeiten.

- **Hilfsfunktionen**

  Diese Kategorie von Ausdrucksfunktionen bietet Hilfsmethoden für spezielle Anforderungen.

- **Numerische Funktionen**

  Diese Kategorie von Ausdrucksfunktionen wurde entwickelt, um mit numerischen Werten umzugehen.

- **Fragebogenfunktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um auf Daten von Fragebögen zuzugreifen. Durch das Benutzen dieser Funktionen kann mithilfe von Ausdrücken auf Kundenantworten zugegriffen werden.

- **Report Viewer-Funktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, um URLs zur Report Viewer-Webanwendung zu generieren.

- **RTF-Funktionen**

  Diese Kategorie von Ausdrucksfunktionen wird verwendet, wenn mit RTF-Nachrichtenvorlagen (Rich-Text-Format) gearbeitet wird.

- **Auswahlfunktionen**

  Diese Kategorie von Ausdrucksfunktionen adressiert Auswahlen.

- **Zeichenfolgenfunktionen**

  Diese Kategorie von Ausdrucksfunktionen wurde konzipiert, um Zeichenfolgen zu handhaben.

- **Systemfunktionen**

  Diese Kategorie von Ausdrucksfunktionen enthält Systemebenenfunktionen.

- **Benutzerdefinierte Funktionen** Hierbei handelt es sich um *benutzerdefinierte Funktionen*, die von Benutzern in Dialogue Admin hinzugefügt wurden. Weitere Informationen finden Sie unter **Ausdrucks-Plug-Ins**.

Aufrufen von Ausdrucksfunktionen

Bitte beachten Sie beim Aufrufen einer Funktion, die mehrere Argumente benötigt, dass die Argumente durch *Kommata getrennt werden müssen.*

Benutzerdefinierte Ausdrucksfunktionen

Es ist möglich, neue Ausdrucksfunktionen zu definieren, indem *Ausdrucks-Plug-Ins* implementiert werden. Solche Funktionen heißen *Benutzerdefinierte Funktionen*. Weitere Informationen finden Sie unter **Ausdrucks-Plug-Ins**.

Aggregatfunktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen unterstützt die Aggregation von Daten.

float **Max** (*array of integer or float* Values)

Gibt den Maximalwert eines Arrays von Werten zurück, welcher der höchste Wert eines Feldes in einer 1:n-Gruppe ist.

Das folgende Beispiel gibt die größte Bestellung eines Kunden zurück.

Beispiel: Max(Orders.Amount)

float **Min**(*array of integer or float* Values)

Gibt den Minimalwert eines Arrays von Werten zurück, welcher der niedrigste Wert eines Feldes in einer 1:n-Gruppe ist.

Das folgende Beispiel gibt die kleinste Bestellung eines Kunden zurück.

Beispiel:Werte für Min(Orders.Amount) sind *unterschiedlich*.

integer **NumberOf**(*datagroup* DataGroup)

Gibt die Anzahl der Elemente in einer Datengruppe für den jeweiligen Kunden zurück.

Das folgende Beispiel gibt die Anzahl der Antwortformulare zurück, die für den Kunden registriert sind.

Beispiel: NumberOf(AnswerForms)

float **Sum**(*array of integer or float* Values)

Gibt die Summe eines Arrays von Werten zurück, welche die Summe eines Feldes in einer 1:n-Gruppe ist.

Das folgende Beispiel gibt die Summe der Beträge aller Bestellungen eines Kunden zurück.

Beispiel: Sum(Orders.Amount)

Analytics-Funktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen unterstützt die Integration mit Portrait Customer Analytics.

float **EvaluateRule**(*string* RuleName)

Wertet das Ergebnis einer Analyseregel aus und gibt es im Zusammenhang mit dem aktuellen Kunden zurück. Die angegebene Regel wird unter Verwendung eines Echtzeitbewertungsmoduls ausgewertet. Der zurückgegebene Wert ist ein Gleitkommawert, was bedeutet, dass die auszuwertende Regel einen numerischen Wert (ganzzahlig oder Gleitkomma) zurückgeben muss.

RuleName ist der Name der Regel.

Beispiel: Evaluaterule("MyScoringRule")

datetime **EvaluateRuleDatetime**(*string* RuleName)

Bewirkt dasselbe wie *EvaluateRule(...)*, gibt aber einen DateTime-Wert zurück. Daher sollte die angegebene Regel ebenfalls einen DateTime-Wert zurückgeben.

string **EvaluateRuleString**(*string* RuleName)

Bewirkt dasselbe wie *EvaluateRule(...)*, gibt aber als Wert eine Zeichenfolge zurück. Daher sollte die angegebene Regel ebenfalls eine Zeichenfolge zurückgeben.

Inhaltsobjektfunktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um Inhaltsobjekte auszuführen.

string ***ContentObject***(*string* ContentObjectKey, *string* RequestParams)

Führt ein Inhaltsobjekt aus, und gibt seinen Inhalt im HTML-Format zurück.

ContentObjectKey ist eine eindeutige Zeichenfolge, die das Inhaltsobjekt identifiziert. Diesen Wert findet man unter „Inhaltsobjekteigenschaften“ in Visual Dialogue.

RequestParams ist eine Zeichenfolge, die zusätzliche Parameter beim Aufrufen der Inhaltsobjekte enthält. Das Format dieser Zeichenfolge ist:

```
<param1>=<value1>;<param2>=<value2>;...;<paramN>=<valueN>
```

Beispiel:

```
MyInfo1=B;MyInfo2=C
```

string ***ContentObjectText***(*string* ContentObjectKey, *string* RequestParams)

Führt ein Inhaltsobjekt aus, und gibt seinen Inhalt als Text zurück.

ContentObjectKey ist eine eindeutige Zeichenfolge, die das Inhaltsobjekt identifiziert. Diesen Wert findet man unter „Inhaltsobjekteigenschaften“ in Visual Dialogue.

RequestParams ist eine Zeichenfolge, die zusätzliche Parameter beim Aufrufen der Inhaltsobjekte enthält. Das Format dieser Zeichenfolge ist:

```
<param1>=<value1>;<param2>=<value2>;...;<paramN>=<valueN>
```

Beispiel:

```
MyInfo1=B;MyInfo2=C
```

Hinweis: Die Funktion **ContentObject(...)** eignet sich, um E-Mails im HTML-Format zu erstellen. **ContentObjectText(...)** eignet sich, um E-Mails in Textform und SMS-Nachrichten zu erstellen.

Datums- und Zeitfunktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um mit Datums- und Zeitwerten zu arbeiten.

datetime **AddMonths**(*datetime* or *date* DateValue, *integer* MonthsToAdd)

Gibt das Datum und die Uhrzeit oder das Datum als DateValue mit zusätzlichen MonthsToAdd-Monaten zurück.

integer **Age**(*datetime* or *date* Value)

Berechnet die Anzahl von Jahren zu einem gegebenen date- oder datetime-Wert und gibt diese zurück. Dies ist sehr bequem, wenn Sie das Alter aus einem gegebenen Geburtsdatum herleiten möchten.

date **DateValue**(*datetime* Value)

Gibt nur den date-Teil des eingegebenen Wertes zurück.

integer **Day**(*datetime* or *date* Value)

Gibt den Tag eines datetime- oder date-Wertes zurück.

integer **DayOfWeek**(*datetime* or *date* Value)

Gibt den Wochentag zu einem datetime- oder date-Wert zurück. Das Ergebnis ist immer eine ganze Zahl zwischen 1 und 7, jedoch legt Ihre Datenbank fest, welcher Tag als erster Tag der Woche betrachtet wird.

datetime **EncodeDateTime**(*string* DatePart, *string* TimePart)

Gibt den datetime-Wert entsprechend den beiden Zeichenfolgenparametern (DatePart und TimePart) zurück. Das Format dieser Parameter folgt den aktuellen regionalen Einstellungen von Dialogue Server.

integer **Month**(*datetime* or *date* Value)

Gibt den Monat zu einem datetime- oder date-Wert zurück.

datetime **Now**()

Gibt Systemdatum und -zeit zurück.

date **Today**()

Gibt das aktuelle Datum ohne den Zeitteil zurück.

integer **Year**(*datetime* or *date* Value)

Gibt das Jahr zu einem datetime- oder date-Wert zurück.

Customer Web Access-Funktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um Kunden beim Zugriff auf Webanwendungen zu unterstützen. Einige der Funktionen erzeugen URLs zu personalisierten Webseiten. Diese Funktionen werden häufig in Nachrichtenvorlagen verwendet.

string **AnswerFormURL**(*integer* UrlType, *integer* QuestionnaireID, integer LayoutID, *integer* BroadcastID, *integer* DialogID, *bool* Scramble)

Gibt eine Zeichenfolge zurück, welche die URL zu einer Antwortformular-Webseite enthält. Diese URL enthält Informationen, die mit dem Antwortformular eines bestimmten Kunden oder Dialogteilnehmers verbunden sind.

UrlType ist eine Zahl, die den Typ der zu generierenden URL anzeigt.

URL-Typindex	Beschreibung
0	Anonyme Antwort
1	Identifizierte Antwort (Kunden-ID in der URL)
2	Kundenanmeldung (unter Verwendung von Anmelde-ID und Kennwort)
3	Teilnehmeranmeldung (unter Verwendung von Teilnehmer-ID und Kennwort)

QuestionnaireID ist die ID des Fragebogens. *LayoutID* ist die ID des anzuzeigenden Layouts. Setzen Sie *LayoutID* auf „-1“, um ein automatisch generiertes Standardlayout zu verwenden.

BroadcastID ist die ID der Übertragung, mit der Antwortformulare verbunden werden sollen. Setzen Sie *BroadcastID* auf „-1“, wenn keine Übertragung verwendet wird.

DialogID ist die ID des Dialogs, wenn die Teilnehmeranmeldung verwendet wird. Setzen Sie *DialogID* auf „-1“, wenn der Dialog nicht relevant ist, oder wenn die Anmeldung unabhängig vom Dialog sein soll.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

In einer E-Mail-Vorlage generiert das folgende Beispiel eines Seriendruckfeldes so eine URL:

```
«#AnswerFormURL(1048, 1101, -1, -1, TRUE)»
```

Hinweis: Das Zeichen „**#**“ teilt dem System mit, dass das Seriendruckfeld als Ausdruck und nicht als einfaches Datenfeld wie „Vorname“ behandelt werden soll.

string **AnswerFormURLEx**(*integer* UrlType, *integer* QuestionnaireID, integer LayoutID, *integer* BroadcastID, *integer* DialogID, *bool* Scramble, *bool* NoTrack)

Bewirkt dasselbe wie *AnswerFormURL(),* enthält aber einen Parameter, der das Nachverfolgen von Antworten zu Testzwecken deaktiviert.

NoTrack bestimmt, ob der Parameter „notrack“ (nicht nachverfolgen) in der generierten URL enthalten ist. Dies deaktiviert die Antwortnachverfolgung in der Anwendung „Customer Web Access“.

Hinweis: Die Option *NoTrack* ist nur nützlich, wenn die Antwortnachverfolgung im verwendeten Fragebogen aktiviert ist.

string **CustLoginID**( )

Gibt die Anmelde-ID des aktuellen Kunden zurück.

string **CustomURL**(*string* BaseURL, *bool* Scramble)

Gibt eine Zeichenfolge zurück, welche die URL zu einer Webseite enthält.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

string ***CustPassword***( )

Gibt das Kennwort des aktuellen Kunden zurück.

string **EmailMessageUrl**(*string* MessageID, *bool* Scramble)

Gibt eine URL zur Anzeige einer E-Mail-Nachricht in der Anwendung „Customer Web Access“ zurück.

MessageID ist die eindeutige Kennung der Nachricht.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

string **EmailTrackImage**( )

Diese Funktion wird in HTML-E-Mails verwendet und gibt eine Zeichenfolge zurück, die ein HTML-Bild-Tag enthält. Dieses Bild-Tag lädt ein Bild aus der Anwendung „Web Utilities“ herunter. Diese Anwendung wird verfolgen und speichern, dass das Bild heruntergeladen wurde. So wird Dialogue Server wissen, dass ein bestimmter Kunde eine E-Mail zum Lesen geöffnet hat.

string **EmailTrackImageEx**(*string* ImageName)

EmailTrackImageEx bewirkt dasselbe wie *EmailTrackImage*, der Dateiname des Bildes zum Herunterladen kann jedoch durch den Benutzer angegeben werden.

ImageName ist der Dateiname des herunterzuladenden Bildes. Das Bild wird in einem bestimmten Ordner in der Anwendung „Web Utilities“ gespeichert. Es werden nur Bilder im gif-Format unterstützt.

bool **HasOpenedEmail**(*integer* TemplateID)

HasOpenedEmail wird zusammen mit der Funktion *E-Mail-Nachverfolgung* in Dialogue Server verwendet. Wenn einem Kunden eine HTML-E-Mail-Nachricht gesendet wird und „E-Mail-Nachverfolgung“ in dieser Nachricht aktiviert ist, wird das System nachverfolgen, wann der Kunde diese E-Mail öffnet. *HasOpe-*

nEmail wird als *true* zurückgeben, wenn der Kunde eine E-Mail basierend auf einer angegebenen Nachrichtenvorlage öffnet.

TemplateID ist die eindeutige ID einer Nachrichtenvorlage, die in Visual Dialogue entworfen wurde (entsprechend einem Wert in der Datenbankspalte DOC_BASE_MESSAGE.DBM_ID).

bool **HasOpenedLink**(*integer* TemplateID*, string* LinkName)

HasOpenedLink wird zusammen mit der Funktion *Verknüpfungsnachverfolgung* in Dialogue Server verwendet. Eine E-Mail-Nachricht kann Verknüpfungen (URLs) enthalten, für welche die Verknüpfungsnachverfolgung aktiviert ist. Das System verfolgt, wenn Kunden diese Verknüpfungen in den E-Mails öffnen, die sie erhalten haben. *HasOpenedLink* gibt *true* zurück, wenn der Kunde eine bestimmte Verknüpfung in einer angegebenen Nachricht geöffnet hat.

TemplateID ist die eindeutige ID einer Nachrichtenvorlage, die in Visual Dialogue entworfen wurde (entsprechend einem Wert in der Datenbankspalte DOC_BASE_MESSAGE.DBM_ID). Wenn *TemplateID* auf 0 gesetzt ist, wird die Funktion die Prüfung für alle Nachrichten (vorlagenunabhängig) ausführen.

LinkName ist der Name der nachverfolgten Verknüpfung, die beim Entwerfen der Vorlage im Nachrichten-Designer oder beim Aufrufen der Ausdrucksfunktion *TrackURL* angegeben wird.

bool **HasOpenedLinkEx**(*integer* TemplateID*, string Url*)

HasOpenedLinkEx bewirkt genau dasselbe wie *HasOpenedLink*, es wird jedoch als Parameter eine URL anstelle des Namens der nachverfolgten Verknüpfung verwendet.

Url ist die Adresse (URL) der nachverfolgten Verknüpfung.

string **PublicFileURL**(*string* ContentID)

Gibt die URL einer Datei zurück, die für das Internet veröffentlicht wurde. Dateien, die für das Internet veröffentlicht wurden, werden in der Datenbanktabelle WEB_PUBLIC_FILE gespeichert. Die zurückgegebenen URLs verweisen zur Anwendung „Web Utilities“.

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

string **ScrambleURL**(*string* Url)

Verschlüsselt eine URL. *Url* ist die zu verschlüsselnde URL.

string **ShortenAndNameURL**(*string* OriginalUrl, *string* ShortUrlName, *bool* EnableLinkTracking, *bool* ReplaceExistingNamedUrl, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für `OriginalUrl` mit dem optionalen Präfix `ShortUrlPrefix` zurück, das an den benannten Teil des URL-Pfades angefügt wird, z. B. `http://shorturl.pb.com/Sonderangebote/Weihnachten2013`. Wenn `EnableLinkTracking` „true“ ist, werden Informationen zur Verknüpfungsnachverfolgung erfasst und der kurze URL-Name als Verknüpfungsname verwendet. Es wird bei jedem Abruf dieselbe URL zurückgegeben, es sei denn sie wurde vorher mit anderen Parametern abgerufen. In diesem Fall wird ein Fehler ausgegeben. Dieses Verhalten kann überschrieben werden, indem der Parameter `ReplaceExistingNamedUrl` auf „true“ gesetzt wird. Allerdings werden dann die zuvor generierten gekürzten URLs zur letzten `OriginalUrl` weitergeleitet. NB! Die Datenbanktabelle der gekürzten URLs ist für alle Kundendomänen der Instanz dieselbe. Eine benannte gekürzte URL darf nur für eine Domäne verwendet werden.

string **ShortenAndTrackURL**(*string* OriginalUrl, *string* LinkName, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für `OriginalUrl` mit dem optionalen Präfix `ShortUrlPrefix` zurück, das an den codierten Teil der URL angefügt wird, z. B. `http://shorturl.pb.com/Sommerkampagne/AhF56yx`. Portrait Dialogue erfasst Informationen zur Verknüpfungsnachverfolgung für den angegeben `LinkName`. Es wird bei jedem Abruf eine andere URL zurückgegeben.

string **ShortenURL**(*string* OriginalUrl, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für `OriginalUrl` mit dem optionalen Präfix `ShortUrlPrefix` zurück, das an den codierten Teil der URL angefügt wird, z. B. `http://shorturl.pb.com/Sommerkampagne/AhF56yx`. Es wird bei jedem Abruf eine andere URL zurückgegeben.

string **TrackURL (***string* OriginalURL, *string* LinkName)

Wird verwendet, um URLs zu erstellen, die vom Verknüpfungsnachverfolgungssystem nachverfolgt werden. *OriginalURL* ist die URL, die auf die Webseite verweist, zu welcher der Kunde (Browser-Nutzer) geleitet wird. *LinkName* ist der benutzerdefinierte Name der nachzuverfolgenden Verknüpfung.

string **TrackURLAnonymous(** *string* OriginalURL, *string* LinkName)

Wird verwendet, um URLs zu erstellen, die vom Verknüpfungsnachverfolgungssystem nachverfolgt werden. *OriginalURL* ist die URL, die auf die Webseite verweist, zu welcher der Kunde (Browser-Nutzer) geleitet wird. *LinkName* ist der benutzerdefinierte Name der nachzuverfolgenden Verknüpfung. Die Nachverfolgung wird anonym gespeichert, was bedeutet, dass es keine Möglichkeit gibt, den nachverfolgten Kunden zu identifizieren (verwenden Sie für identifiziertes Nachverfolgen die Funktion *TrackURL()*).

string **UnsubscribeURL**(*string* or *integer* CategoryNameOrID, *bool* Scramble)

Gibt eine Zeichenfolge zurück, die eine URL zu einer personalisierten Seite zum Beenden des Abonnements enthält.

CategoryName ist der Name oder die eindeutige ID der Kategorie, die das Abonnement steuert. Eine solche Kategorie kann beispielsweise „Newsletter“ heißen. Alle Kunden, die Mitglieder dieser Kategorie sind, erhalten einen wöchentlichen Newsletter. Über die Webseite, auf welche die URL verweist, können Kunden ihre Mitgliedschaft in dieser Kategorie steuern.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

string **WebProfileLoginURL**(*bool* Scramble)

Gibt eine Zeichenfolge zurück, die eine URL zur Anmeldeseite des Webprofils des Kunden enthält.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

string **WebProfileURL**(*bool* Scramble)

Gibt eine Zeichenfolge zurück, die eine URL zum Webprofil des Kunden enthält.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für den Kunden auszublenden, der die URL öffnet.

Dialogfunktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um mit Dialogen und Dialogteilnehmern zu arbeiten.

string **GetCustomValue**(*string* or *integer* ValueName)

Gibt einen Wert der benutzerdefinierten Wertesammlung eines Kunden zurück. *ValueName* ist der Name des Wertes.

Hinweis: Weitere Informationen finden Sie unter *SetCustomValue* auf den folgenden Seiten.

bool **HasBranchHistory**(*integer* BranchID)

Gibt einen booleschen Wert zurück, der *true* anzeigt, wenn der Kunde durch eine bestimmte Dialogverzweigung behandelt wurde. Dies tritt dann ein, wenn der Kunde an einer Stelle durch die Verzweigung hindurch in die Empfängergruppe der Verzweigung verschoben wurde.

BranchID ist die eindeutige Kennung der Dialogverzweigung. Dieser Wert kann in Dialog Designer von Visual Dialogue im Fenster *Operation bearbeiten* gefunden werden.

Hinweis: Die Teilnehmeranmeldung muss für die jeweilige Dialogoperation aktiviert sein, um diese Funktion nutzen zu können.

bool **HasOperationHistory**(*integer* OperationID)

Gibt einen booleschen Wert zurück, der *true* anzeigt, wenn der Kunde durch eine bestimmte Dialogoperation behandelt wurde. Dies tritt dann ein, wenn der Kunde an einer Stelle durch eine der Verzweigungen der Operation hindurch verschoben wurde.

OperationID ist die eindeutige Kennung der Dialogoperation. Dieser Wert kann in Dialog Designer von Visual Dialogue im Fenster *Operation bearbeiten* gefunden werden.

Hinweis: Die Teilnehmeranmeldung muss für die jeweilige Dialogoperation aktiviert sein, um diese Funktion nutzen zu können.

void **SetCustomValue**(*string* or *integer* ValueName, *string* Value)

Legt einen Wert in der benutzerdefinierten Wertesammlung eines Teilnehmers fest. *ValueName* ist der Name des festzulegenden Wertes.

Benutzerdefinierte Werte sind Werte in einer für Dialogteilnehmer gespeicherten Name/Wert-Sammlung. Für jeden Teilnehmer steht eine beliebige Anzahl von Name/Wert-Paaren zur Verfügung.

Benutzerdefinierte Werte werden im Datenbankfeld DLG_PARTICIPANT.DP_CUSTOM_VALUES gespeichert.

Hinweis: Das Benutzen dieser Funktion außerhalb des Kontexts eines Teilnehmers wird nicht fehlschlagen, aber der Wert wird nicht in der Datenbank gespeichert.

Hilfsfunktionen

Übersicht

Die folgenden Ausdrucksfunktionen bieten Hilfsmethoden für spezielle Anforderungen.

datetime **FieldValueDatetime**(*string* FieldName)

Gibt den Wert eines Datenfeldes in der Kundendomäne oder einer benutzerdefinierten SQL-Anweisung zurück. Benutzerdefinierte SQL-Anweisungen werden im Nachrichten-Designer in Visual Dialogue verwendet. Wenn das angegebene Feld zu einer 1:n-Datengruppe gehört, wird der Wert des aktuellen Datensatzes zurückgegeben.

float **FieldValueNumber**(*string* FieldName)

Gibt den Wert eines Datenfeldes in der Kundendomäne oder einer benutzerdefinierten SQL-Anweisung zurück. Benutzerdefinierte SQL-Anweisungen werden im Nachrichten-Designer in Visual Dialogue verwendet. Wenn das angegebene Feld zu einer 1:n-Datengruppe gehört, wird der Wert des aktuellen Datensatzes zurückgegeben.

string **FieldValueString**(*string* FieldName)

Gibt den Wert eines Datenfeldes in der Kundendomäne oder einer benutzerdefinierten SQL-Anweisung zurück. Benutzerdefinierte SQL-Anweisungen werden im Nachrichten-Designer in Visual Dialogue verwendet. Wenn das angegebene Feld zu einer 1:n-Datengruppe gehört, wird der Wert des aktuellen Datensatzes zurückgegeben.

Numerische Funktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um mit Dialogen und Dialogteilnehmern zu arbeiten.

float **Random**( )

Gibt eine zufällige Zahl zwischen 0 und 1 zurück (0 <= n < 1).

Hinweis: Wenn diese Funktion durch SQL ausgedrückt wird, verwendet sie DBMS-spezifische Funktionen, und die Implementierung unterscheidet sich für SQL Server und Oracle.

Fragebogenfunktionen

Übersicht

Die folgenden Ausdrucksfunktionen werden für den Zugriff auf Fragebögendaten verwendet. Durch das Benutzen dieser Funktionen kann mithilfe von Ausdrücken auf Kundenantworten zugegriffen werden.

Einige Parameter werden für die folgenden Funktionen verwendet und müssen näher erklärt werden:

QuestionaireID ist die ID des Fragebogens, *QuestionKey* ist der Schlüssel der Frage. *QuestionKey* ist eine Zeichenfolge, welche die Frage identifiziert und innerhalb eines Fragebogens eindeutig ist. *QuestionKey* wird einer Frage automatisch zugewiesen, wenn sie im Fragebogen-Designer in Visual Dialogue in einen Fragebogen eingefügt wird. Sie hat folgendes Standardformat: Q1, Q2, Q3, ..., Q*n*. Der Benutzer kann *QuestionKey* jedoch bearbeiten und z. B. den Schlüssel „VNAME“ für die Frage „Vorname“ verwenden.

QuestionKey wird auf Fragen angewendet. Für Alternativen in einer Frage verwendet man *AlternativeKey*. *AlternativeKey* ist innerhalb einer Frage eindeutig. Das Standardformat für „AlternativeKey“ ist: A1, A2, A3, ..., A*n*.

IncludeIncomplete ist ein boolescher Parameter, der festlegt, ob unvollständige Antwortformulare bei der Auswertung des Ergebnisses verschiedener Funktionen zählen. Wenn dieser Parameter ausgelassen wird, ist sein Standardwert „false“, so dass nur vollständige Antwortformulare berücksichtigt werden.

bool **AnswerCombination**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKeys, *bool* UseOR, *bool* IncludeIncomplete = False)

AnswerCombination gibt abhängig von der Antwortenkombination *true* oder *false* zurück.

AlternativeKeys ist eine durch Semikola getrennte Liste der Alternativschlüssel.

UseOR zeigt an, ob alle Alternativen in *AlternativeKeys* beantwortet wurden oder nur eine davon.

Beispiel: AnswerCombination(1000, "Q2", "A1;A2;A3", True)

string **AnswerComment**(*integer* QuestionnaireID, *integer* SectionNo, *bool* IncludeIncomplete = False)

Gibt den Kommentar des angegebenen Abschnitts zurück.

SectionNo ist die Nummer des Abschnitts, beginnend mit „1“ für den ersten Abschnitt des Fragebogens.

date **AnswerDate**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt die Antwort einer Frage oder Alternative des Datentyps „date“ zurück.

date **AnswerDateTime**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt die Antwort einer Frage oder Alternative des Datentyps „datetime“ zurück.

date **AnswerFloat**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt die Antwort einer Frage oder Alternative des Datentyps „float“ zurück.

date **AnswerInt**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt die Antwort einer Frage oder Alternative des Datentyps „integer“ zurück.

string **AnswerSingleChoice**(*integer* QuestionnaireID, *integer* QuestionKey, *bool* IncludeIncomplete = False)

Gibt den Alternativschlüssel einer Antwort in einer Einzelauswahlfrage zurück.

string **AnswerSingleChoiceText**(*integer* QuestionnaireID, *integer* QuestionKey, *bool* IncludeIncomplete = False)

Gibt die Beschriftung der Alternative der Antwort in einer Einzelauswahlfrage zurück.

date **AnswerText**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt die Antwort einer Frage oder Alternative des Datentyps „string“ zurück.

bool **HasAnswerForm**(*integer* QuestionnaireID, *bool* IncludeIncomplete = False)

Gibt *true* zurück, wenn der Kunde den angegebenen Fragebogen (im jeweiligen Kontext) beantwortet hat. Anderenfalls wird *false* zurückgegeben.

bool **HasInclompleteResponse**(*integer* QuestionnaireID, *integer* LayoutID, *integer* PageIndex)

Gibt *true* (wahr) zurück, wenn der Kunde angefangen hat, den angegebenen Fragebogen zu beantworten, die Beantwortung aber nicht abgeschlossen hat. *HasIncompleteResponse( )* kann nur vorliegen, wenn *Antwortnachverfolgung* aktiviert ist.

LayoutID ist die eindeutige ID des verwendeten Fragebogenlayouts.

PageIndex ist die Nummer der eingegebenen Seite (d. h. der dem Antwortenden angezeigten Seite) in der unvollständigen Antwort.

PageIndex beginnt mit 1 auf der ersten Seite des Layouts.

bool **IsAnswered**(*integer* QuestionnaireID, *string* QuestionKey, *string* AlternativeKey, *bool* IncludeIncomplete = False)

Gibt *true* zurück, wenn eine bestimmte Frage oder Alternative beantwortet (oder bei Mehrfachauswahlfragen ausgewählt) wurde. Anderenfalls wird *false* zurückgegeben.

Report Viewer-Funktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen wird verwendet, um URLs zur Report Viewer-Webanwendung zu generieren.

string **ArchivedReportURL**(*string* or *integer* ArchivedReportID, *bool*Scramble)

Gibt eine Zeichenfolge zurück, welche die URL zum Anzeigen des angegebenen, archivierten Berichts enthält. Der Bericht wird im Standardanzeigeformat für archivierte Berichte angezeigt.

ArchivedReportID ist die eindeutige ID des archivierten Berichts (Datenbankspalte REPORT_ARCIVDE.RA_ID).

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für die Person auszublenden, welche die URL öffnet.

string ***ArchivedReportURL***(*string* or *integer* ArchivedReportID, *integer* FormatIndex, *integer* Action, *bool* Scramble)

Gibt eine Zeichenfolge zurück, welche die URL zum Anzeigen des angegebenen, archivierten Berichts im festgelegten Format enthält.

ArchivedReportID ist die eindeutige ID des archivierten Berichts (Datenbankspalte REPORT_ARCIV-DE.RA_ID).

FormatIndex ist ein ganzzahliger Wert, der das verwendete Format zum Anzeigen des archivierten Berichts angibt. Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

Action ist ein ganzzahliger Wert, der angibt, wie der Bericht dem Benutzer dargestellt wird. Eine Liste der Aktionsdefinitionen finden Sie unter **Berichtsanzeigeaktionen**.

Scramble bestimmt, ob die URL-Parameter verschlüsselt werden sollen, um ihre Inhalte für die Person auszublenden, welche die URL öffnet.

RTF-Funktionen

Übersicht

Die folgenden Ausdrucksfunktionen werden verwendet, wenn mit RTF-Nachrichtenvorlagen (Rich-Text-Format) gearbeitet wird.

string **RtfInsertPicture**(*string* Filename, *integer* ScaleFactor)

Fügt ein Bild in eine RTF-Vorlage ein, während die Nachricht zusammengefügt wird. Der zurückgegebene Wert ist eine Zeichenfolge, welche die RTF-Bildmarkierung darstellt.

Filename ist der volle Pfad und Dateiname der einzufügenden Bilddatei.

ScaleFactor ist eine Zahl, die anzeigt, ob das Bild skaliert werden soll.

ScaleFactor wird in Prozent der Originalgröße des Bildes angegeben. Der Wert „0“ (oder „100“) bedeutet keine Skalierung.

string **RtfInsertPictureCID**(*string* CID, *integer* ScaleFactor)

RtfInsertPictureCID bewirkt dasselbe wie *RtfInsertPicutre*, das Bild wird jedoch aus „Veröffentlichte Dateien“ geladen und nicht aus einer Datei.

CID ist die eindeutige Kennung in der Bibliothek von „Veröffentlichte Dateien“ in Dialogue Server.

Hinweis: *Veröffentlichte Dateien* werden in der Hilfedatei von Visual Dialogue beschrieben.

Auswahl- und Listenfunktionen

Übersicht

Diese Kategorie von Ausdrucksfunktionen gilt für Auswahlen und Listen.

bool **IsInSelection**(*integer* SelectionID)

Gibt *true* zurück, wenn der Kunde in der durch *SelectionID* angegebenen Auswahl enthalten ist.

IsInSelection wird verwendet, um Unterauswahlkriterien im Auswahldesigner zu definieren.

Hinweis: Bei der Verwendung von *IsInSelection* gibt es zwei Beschränkungen: 1) Die durch *SelectionID* angegebene Auswahl kann keine Kontextwerte angeben („Kontext in Auswahl verwenden“ kann im Auswahldesigner nicht aktiviert werden). 2) Die Auswahl muss in eine SQL-Anweisung konvertierbar sein (dies wird von den meisten Auswahlen erfüllt).

bool **IsInList**(*integer* or *string* List)

Gibt *TRUE* zurück, wenn der Kunde in der durch *List* angegebenen Liste enthalten ist.

Der *List*-Parameter kann entweder die eindeutige ID der Liste (Datenbankspalte `LIST.LST_ID`) enthalten oder den Namen der Liste (Datenbankspalte `LIST.LST_NAME`).

IsInList kann beispielsweise verwendet werden, um im Auswahldesigner Kriterien für eine Liste von Kunden zu definieren, die aus der Portrait Explorer-Anwendung exportiert wurde.

Zeichenfolgenfunktionen

Übersicht

Die folgenden Ausdrucksfunktionen wurden zum Umgang mit Zeichenfolgewerten entwickelt.

string **HtmlEncode**(*string* UncodedString)

Gibt die eingegebene Zeichenfolge HTML-codiert zurück.

string **IfString**(*bool* Condition*, string* TrueValue*, string* FalseCondition)

Gibt eine Zeichenfolge zurück, wenn „Condition“ wahr ist, und eine andere, wenn nicht.

Im folgenden Beispiel ändert sich der zurückgegebene Wert abhängig davon, ob eine Kommunikation über SMS-Nachrichten stattgefunden hat oder nicht.

Beispiel: IfString(has Activities[Channel="SMS"], "hat SMS erhalten", "hat SMS nicht erhalten")

integer **Length**(*string* Value)

Gibt die Länge der eingegebenen Zeichenfolge zurück.

string **LowerCase**(*string* Value)

Gibt die eingegebene Zeichenfolge in Kleinbuchstaben zurück.

integer **StrToInt** (*string* Value)

Konvertiert eine Zeichenfolge in eine ganze Zahl. Der eingegebene Wert muss eine gültige ganze Zahl sein. Anderenfalls wird eine Ausnahme erzeugt.

string **ToString**(Value)

Konvertiert einen eingegebenen Wert, der keine Zeichenfolge ist, in eine Zeichenfolge.

string **UpperCase**(*string* Value)

Gibt die eingegebene Zeichenfolge in Großbuchstaben zurück.

Systemfunktionen

Übersicht

Die folgenden Ausdrucksfunktion sind Funktionen für die Systemebene.

string **UserCellular**()

Gibt die Mobiltelefonnummer des aktuellen Benutzers zurück.

string **UserDisplayName**()

Gibt den echten Namen des aktuellen Benutzers zurück.

string **UserEmail**()

Gibt die E-Mail-Adresse des aktuellen Benutzers zurück.

string **UserName**()

Gibt den Benutzernamen des aktuellen Benutzers zurück.

Kapitel

3

Einrichten von Berichten

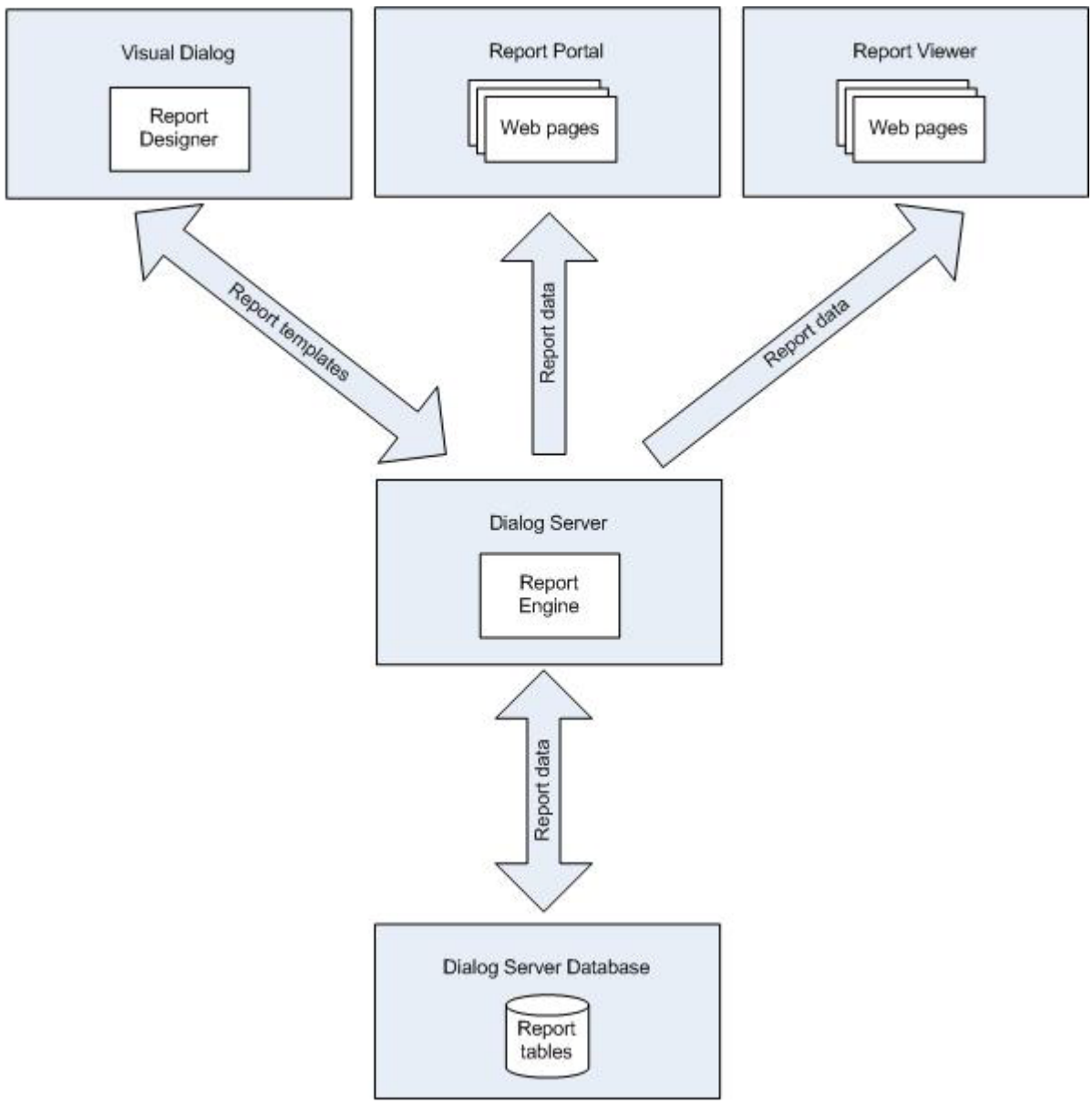
In diesem Abschnitt:

- Berichte 80
- Berichtanzeige 81
- Berichtsformate 83
- Berichtsparameter-XML 84
- Berichtsansichtsaktionen 86

Berichte

Info über Berichte

Portrait Dialogue stellt Möglichkeiten zur Berichtserstellung zur Verfügung. Verschiedene Module der Produktfamilie enthalten unterschiedliche Teile der Berichtsfunktion. Die unten stehende Grafik stellt dies dar.



Report Designer von Visual Dialogue

Berichte werden in Visual Dialogue erstellt. In Report Designer arbeitet der Benutzer mit Berichtsvorlagen und definiert Datenquellen, Parameter und das Layout eines Berichts.

Report Engine von Dialogue Server

Report Engine ist innerhalb von Dialogue Server dafür verantwortlich, dass Berichte in unterschiedlichen Formaten ausgegeben werden. Mithilfe der Dialogue Server API (ReportAPI) können Programmierer und Integratoren mit Berichten programmatisch arbeiten. Definitionen für Berichtsvorlagen und archivierte Berichte werden in der Datenbank von Dialogue Server in einem Tabellensatz gespeichert.

Report Portal

Report Portal ist eine Webanwendung, die Benutzern eine Schnittstelle bereitstellt, um Berichte zu durchsuchen, auszuführen und anzuzeigen. Benutzer melden sich bei dieser Anwendung an, um Berichte entsprechend ihrer Zugriffsrechte auszuführen und anzuzeigen.

Berichtanzeige

Report Viewer ist eine Webanwendung, welche die Verteilung von Berichten unterstützt. Report Viewer unterstützt außerdem das Anzeigen und Herunterladen von Berichten über URL-Parameter. Benutzer melden sich bei Report Viewer nicht an, sondern erhalten nur auf einen einzelnen, in einer URL angegebenen Bericht Zugriff. Diese URLs sind in der Regel verschlüsselt und enthalten eine Prüfsumme, um Manipulationen zu vermeiden. **Weitere Infos**.

Berichtanzeige

Einleitung

Report Viewer wird verwendet, um archivierte Berichte anzuzeigen und Berichtsvorlagen auszuführen.

Mit einem archivierten Bericht verknüpfen

Es ist möglich, eine direkte Verknüpfung zu einem archivierten Bericht zu erstellen. Dadurch wird der archivierte Bericht in Report Viewer geöffnet.

Das beschriebene URL-Format:

(**Fetter Text** stellt optionale Parameter dar.)

- http://<webserver>/MHReportViewer/ReportViewer.aspx?

  ArchiveID=[ARCHIVE ID]**&Action=[ACTION]&**

  FormatIndex=[FORMAT INDEX]&Instance=[INSTANCE_NAME]

URL-Parameter:

- Der Parameter „ArchiveID“ ist erforderlich. Ersetzen Sie [Archiv-ID] in der URL durch die archivierte Berichts-ID.
- Der Parameter „Action“ ist optional. Verwenden Sie diesen Parameter, um verschiedene Ansichtsaktionen für den Bericht festzulegen. Ersetzen Sie [ACTION] in der URL mit der gewünschten Aktion. Weitere Informationen finden Sie unter **Berichtsansichtsaktionen**.
- Der Parameter „FormatIndex“ ist optional. Verwenden Sie diesen Parameter, um das Standardformat des archivierten Berichts zu überschreiben. Ersetzen Sie [FORMATINDEX] in der URL mit dem gewünschten Format. Weitere Informationen finden Sie unter **Berichtsformate**.
- Der Parameter „Instance“ ist optional. Verwenden Sie diesen Parameter, um den zu verwendenden Instanznamen zu spezifizieren. Wird dieser nicht spezifiziert, wird die Standardinstanz verwendet.

Beispiel:

- http://<webserver>/MHReportViewer/ReportViewer.aspx?

  ArchiveID=1061&FormatIndex=1&Action=Browse

Ausführen von Berichtsvorlagen

Es ist möglich, eine Verknüpfung zu Report Viewer zu erstellen, und einen Bericht ohne Parameter auszuführen.

Wenn ein Bericht mit Berichtsparametern ausgeführt wird, müssen die Vorlagen-ID und die Berichtsparameter angegeben werden. Bei der Angabe der Berichtsparameter in der URL müssen der Parametername und der Parameterwert (und optional der Parameteroperator) angegeben werden. Falls der Berichtsparameteroperator nicht angegeben wird, wird der Standardoperator verwendet.

Das beschriebene URL-Format:

(**Fetter Text** stellt optionale Parameter dar.)

- Verknüpfen ohne Berichtsparameter:

  http://<webserver>/MHReportViewer/ReportViewer.aspx?

  TemplateID=[TEMPLATE ID]**&Action=[ACTION]&FormatIndex=[FORMAT INDEX]**

- Mit Berichtsparametern:

  http://<webserver>/MHReportViewer/ReportViewer.aspx?

  TemplateID=[TEMPLATE ID]**&Action=[ACTION]&FormatIndex=[FORMAT INDEX]&Instance=[INSTANCE_NAME]&**

  pname[1,2...]=[PARAMETER NAME]&

  pvalue[1,2...]=[PARAMETER-VALUE]&

  poperator[1,2...]=[PARAMETER-OPERATOR]&

URL-Parameter:

- Der Parameter „TemplateID“ ist erforderlich. Ersetzen Sie [TEMPLATE ID] in der URL durch die Vorlagen-ID des Berichts.

- Der Parameter „Action“ ist optional. Verwenden Sie diesen Parameter, um verschiedene Ansichtsaktionen für den Bericht festzulegen. Ersetzen Sie [ACTION] in der URL mit der gewünschten Aktion. Weitere Informationen finden Sie unter **Berichtsansichtsaktionen**.
- Der Parameter „FormatIndex“ ist optional. Verwenden Sie diesen Parameter, um das Standardformat des archivierten Berichts zu überschreiben. Ersetzen Sie [FORMATINDEX] in der URL mit dem gewünschten Format. Weitere Informationen finden Sie unter **Berichtsformate**.
- Der Parameter „Instance“ ist optional. Verwenden Sie diesen Parameter, um den zu verwendenden Instanznamen zu spezifizieren. Wird dieser nicht spezifiziert, wird die Standardinstanz verwendet.
- Geben Sie den ersten Berichtsparameter wie folgt an: `pname1=[PARAMETER NAME]&pvalue1=[PARAMETER VALUE]`

  `Geben Sie den zweite Berichtsparameter wie folgt an: pname2=[PARAMETER NAME]&pvalue2=[PARAMETER VALUE]`

  und so weiter.

Beispiel:

- http://<webserver>/MHReportViewer/ReportViewer.aspx?

  TemplateID=1267&FormatIndex=1&Action=Browse

Verschlüsselte URLs

In den obigen Beispielen werden die URL-Parameterwerte (z. B. „ArchiveID“) in Klartext angegeben. Report Viewer unterstützt auch URL-Parameterwerte, die mit einer Prüfsumme verschlüsselt sind. Dialogue Admin enthält ein Werkzeug mit dessen Hilfe Sie URLs ver- und entschlüsseln können. Dieses Werkzeug ist im Hauptmenü von Dialogue Admin unter *Tools* verfügbar.

Beispiel:

Nicht verschlüsselte URL: http://<webserver>/MHReportViewer/ReportViewer.aspx?

ArchiveID=1061&FormatIndex=1&Action=Browse

Verschlüsselte URL: http://<webserver>/MHReportViewer/ReportViewer.aspx?

ArchiveID=085D0158&FormatIndex=08&Action=7B1F581E4609&CheckSum=015A0F

Berichtsformate

Einleitung

Es steht eine Reihe von Formaten zur Anzeige von Berichten zur Verfügung. Ein *Formatindex* wird verwendet, um die Formate anzugeben, wenn die Dialogue Server-API aufgerufen wird, oder wenn mit URLs gearbeitet wird, um auf Report Portal und die Report Viewer-Webanwendungen zuzugreifen.

Übersicht über Berichtsformate

In nachstehender Tabelle werden die verschiedenen Formate definiert.

Formatin-dex	Beschreibung	Ausgabedateien
0	Adobe Acrobat PDF-Dateien.	Es wird eine PDF-Datei erzeugt.
1	HTML - mit Seitenumbrüchen	Für jede Seite wird eine HTML-Datei erzeugt. Bilder und Diagramme werden als JPG-Dateien ausgegeben.
2	HTML - eine Seite	Es wird eine HTML-Datei mit allen Seiten erzeugt. Bilder und Diagramme werden als JPG-Dateien ausgegeben.
3	XHTML	Für jede Seite wird eine XHTML-Datei erzeugt. Bilder und Diagramme werden als JPG-Dateien ausgegeben.
4	Rich-Text-Format	Es wird eine RTF-Datei erzeugt.
5	Microsoft Excel – einzelne Seite	Es wird ein Excel-Arbeitsblatt erzeugt. Es enthält keine Grafiken.
6	JPEG-Bildformat	Für jede Seite wird eine JPG-Datei erzeugt.
7	GIF-Bildformat	Für jede Seite wird eine GIF-Datei erzeugt.
8	BITMAP-Bildformat	Für jede Seite wird eine BMP-Datei erzeugt.
9	Windows Metafile	Für jede Seite wird eine WMF-Datei erzeugt.
10	Erweiterte Windows-Metadatei	Für jede Seite wird eine EMF-Datei erzeugt.
11	Microsoft Excel - mehrere Seiten	Für jede Berichtsseite wird ein Excel-Arbeitsblatt erzeugt. Enthält Grafiken (z. B. Bilder). Diagramme werden als Bilder eingeschlossen.

Beispiele

Nachstehend finden Sie ein Beispiel einer URL zur Anzeige eines Berichts in Report Viewer:

```
http://<webserver>/MHReportViewer/ReportViewer.aspx?
ArchiveID=1061&FormatIndex=1&Action=Browse
```

Berichtsparameter-XML

Übersicht

Die Berichtsparameter-XML ist ein XML-Dokument, das in der Berichts-API von Dialogue Server genutzt wird. Sie wird verwendet, um Berichtsparameter anzugeben, die beim Erstellen eines Berichts eingesetzt wurden.

Format der Berichtsparameter-XML

Wenn Sie einen Parameter in der Berichtsparameter-XML angeben, werden der Name des Parameters, der Datentyp, der Operator und der Wert angegeben.

- Jeder Parameter wird durch einen XML-Knoten **param** dargestellt.
- Alle Knoten **param** sind Nachfolger des Stamm-XML-Knotens **xml**.
- Jeder XML-Knoten **param** gibt einen Parameternamen, den Datentyp, den Operator und den Wert an.
  - Geben Sie Zeichenfolgen im Knoten **string_value** an.
  - Geben Sie numerische Werte (ganzzahlige oder Gleitkomma) im Knoten **number_value** an.
  - Geben Sie datetime-Parameterwerte im Knoten **datetime_value** an.
- Gültige Operatoren sind:
  - Gleich (**=**)
  - Kleiner als (**<**)
  - Kleiner als oder gleich (**<=**)
  - Größer als (**>**)
  - Größer als oder gleich (**>=**)
  - Nicht gleich (**<>**)
  - In (**in**) (nur anwendbar für Zeichenfolgen und ganzzahlige Datentypen).

Berichtsparameter-XML – Beispiel

Nachfolgend ist ein Beispiel der Berichtsparameter-XML aufgeführt.

Hinweis: Der Operator muss codiert werden, da ≤ zu <= und ≥ zu >= codiert wird.

```
<xml>
  <param>
    <name>Contact ID</name>
    <type>integer</type>
    <operator>=</operator>
    <number_value>1031</number_value>
  </param>
  <param>
    <name>Contact name</name>
    <type>string</type>
    <operator>=</operator>
    <string_value>John Johnsen</string_value>
  </param>
  <param>
    <name>Last Contacted</name>
    <type>datetime</type>
    <operator>&lt;=</operator>
    <datetime_value>2006-10-06</datetime_value>
  </param>
  <param>
    <name>Scoring</name>
    <type>float</type>
    <operator>&lt;&gt;</operator>
```

```
    <number_value>5.5</number_value>
  </param>

</xml>
```

Verwenden des Operators „IN“

Der Operator „IN“ wird zur Angabe mehrfacher Parameterwerte verwendet. Er funktioniert nur mit den Datentypen *Zeichenfolge* und *Ganzzahl*. Im folgenden Beispiel wird gezeigt, wie der Operator „IN“ verwendet wird:

```
<xml>
 <param>
    <name>address_types</name>
    <type>string</type>
    <operator>in</operator>
    <string_value>'office_address', 'postal_address', 'delivery_ad-
dress'<string_value>
 </param>
 <param>
    <name>Products</name>
    <type>integer</type>
    <operator>in</operator>
    <string_value>2, 44, 80, 34</string_value>
 </param>
</xml>
```

Berichtsansichtsaktionen

Einleitung

Wenn ein Bericht durch Verwendung einer URL in Report Viewer angezeigt wird, können verschiedene Ansichtsaktionen als URL-Parameter angegeben werden.

Übersicht über Berichtsansichtsaktionen

In der nachstehenden Tabelle werden die verschiedenen Aktionen definiert.

Aktionsindex	Aktion	Beschreibung
0	Durchsuchen	Der Bericht wird im Webbrowser mit einer Navigationsleiste angezeigt. Diese Ansichtsaktion ist für Berichte geeignet, die aus mehreren Dateien (z. B. HTML mit Seitenumbrüchen) bestehen. **Hinweis:** Wenn der Bericht ein Format hat, das nicht in einer HTML-Seite angezeigt werden kann, wird die Ansichtsaktion *Open* verwendet. Ansonsten ist „Browse“ die Standardaktion.

1	Öffnen	Die Berichtsdatei wird im Webbrowser geöffnet. Das Standardverhalten zum Öffnen dieses Dateityps wird verwendet. **Hinweis:** Wenn der Bericht aus mehreren Dateien besteht (z. B. HTML mit Seitenumbrüchen), wird standardmäßig die erste Seite geöffnet. Verwenden Sie den Abfragezeichenfolgenparameter *PageNumber*, um eine andere Seite anzugeben (`...Action=Open&PageNumber=3...`). Das Angeben einer nicht existierenden Seite führt zu einer Fehlermeldung.
2	Speichern	Die Berichtsdatei wird heruntergeladen und der Benutzer wird vom Browser aufgefordert, sie an einem Speicherort zu speichern. **Hinweis:** Wenn der Bericht aus mehreren Dateien besteht (z. B. HTML mit Seitenumbrüchen), wird nur die erste Seite heruntergeladen.
3	Drucken	Die Berichtsdatei wird heruntergeladen, und der Browser zeigt den Druckdialog an. **Hinweis:** Diese Ansichtsfunktion funktioniert nur mit einem HTML-Berichtsformat.
4	PrintClose	Bewirkt dasselbe wie *Print*, aber das Browserfenster schließt sich nach dem Drucken (benötigt Benutzerbestätigung, falls nicht von einer anderen Webseite geöffnet). **Hinweis:** Diese Ansichtsfunktion funktioniert nur mit einem HTML-Berichtsformat.

Beispiele

Nachstehend finden Sie ein Beispiel einer URL zur Anzeige eines Berichts in Report Viewer:

```
http://<webserver>/MHReportViewer/ReportViewer.aspx?
ArchiveID=1061&FormatIndex=1&Action=Browse
```

Siehe auch

Ähnliche Ansichtsaktionen werden in Report Portal verwendet. Weitere Details finden Sie in der Dokumentation von Report Portal.

Kapitel 4

E-Mail- und Verknüpfungsnachverfolgung

In diesem Abschnitt:

- Nachverfolgung 90
- E-Mail Nachverfolgung 90
- Verknüpfungsnachverfolgung 91
- E-Mail- und Verknüpfungsnachverfolgung – Details 93
- Antwortnachverfolgung 95

Nachverfolgung

E-Mail- und Verknüpfungsnachverfolgung

Die Funktion **E-Mail-Nachverfolgung** und **Verknüpfungsnachverfolgung** ist in der Anwendung „Web Utilities“ enthalten. Die Nachverfolgung wird hauptsächlich in E-Mails verwendet, die mit Visual Dialogue erstellt wurden (weitere Informationen finden Sie in der Dokumentation von Visual Dialogue). Sie kann aber auch unabhängig verwendet werden, um URLs für die Nachverfolgung für E-Mails und Webseiten hinzuzufügen.

Eine Beschreibung, wie die Nachverfolgung vom System protokolliert wird, und andere technische Details finden Sie unter **E-Mail- und Verknüpfungsnachverfolgung – Details**.

Antwortennachverfolgung

Die Funktion **Antwortnachverfolgung** ist Teil der Fragebogenfunktionen und wird verwendet, wenn Antworten durch die Anwendung „Customer Web Access“ (CWA) oder die Webanwendung „Telemarketing“ registriert werden.

E-Mail Nachverfolgung

Einleitung

Eine Bildkennzeichnung wird in die nachzuverfolgende E-Mail mit eingeschlossen. Wenn der E-Mail-Empfänger die E-Mail öffnet, wird das Bild heruntergeladen, und die Informationen über den Empfänger werden aufgezeichnet.

Bild-URL-Struktur

Die Bild-URL enthält verschiedene Teile, die verwendet werden, um die E-Mail zu identifizieren, die das Bild herunterlädt.

Die URL enthält die folgenden Parameter:

Parameter	Name
Instanz	Dialogue Server-Instanz
Kundennachricht-ID	Diese ID identifiziert den Kunden und die Nachricht. Dieser Wert entspricht der Datenbankspalte CUSTOMER_MESSAGE.CM_ID. Setzen Sie *Kundennachricht-ID* auf „0“, wenn der Wert unbekannt ist, oder auf „nt“, um die Nachverfolgung zu deaktivieren.

Abbildung 1 – Die verschiedenen Teile der Bild-URL.

Nachverfolgungsbild

Das Standardbild zur Nachverfolgung ist ein transparentes 1 × 3 Pixel (drei Pixel breites) Bild im gif-Format. Da es transparent ist, wird es in der E-Mail nicht zu sehen sein. Aufgrund seiner geringen Größe von 1 × 3 Pixeln hat es minimalen Einfluss auf die anderen E-Mail-Elemente.

Das Standardbild kann mit anderen Bildern ausgetauscht werden, indem diese im Ordner der Anwendung „Web Utilities“ dem Ordner „Resources/ETImages“ hinzugefügt werden. Das neue Bild wird verwendet, wenn das Standardbild im Teil *Heruntergeladenes Bild* der Bild-URL (in Abbildung 1 gezeigt) durch das neue Bild ersetzt wird. Wenn kein Bild im Katalog „ETImages“ gefunden wird, das dem angegebenen Namen entspricht, wird das Standardbild verwendet.

Verknüpfungsnachverfolgung

Einleitung

Die Verknüpfungsnachverfolgung wird verwendet, wenn ein Benutzer auf eine Verknüpfung klickt, für welche die Nachverfolgung aktiviert ist. Wenn ein Benutzer auf eine nachverfolgte Verknüpfung klickt, wird er zuerst zum Teil für die Verknüpfungsnachverfolgung der Anwendung „Web Utilities“ geleitet. Die Anwendung „Web Utilities“ protokolliert den Klick und leitet den Benutzer dann zur Ziel-URL um.

Verknüpfungs-URL-Struktur

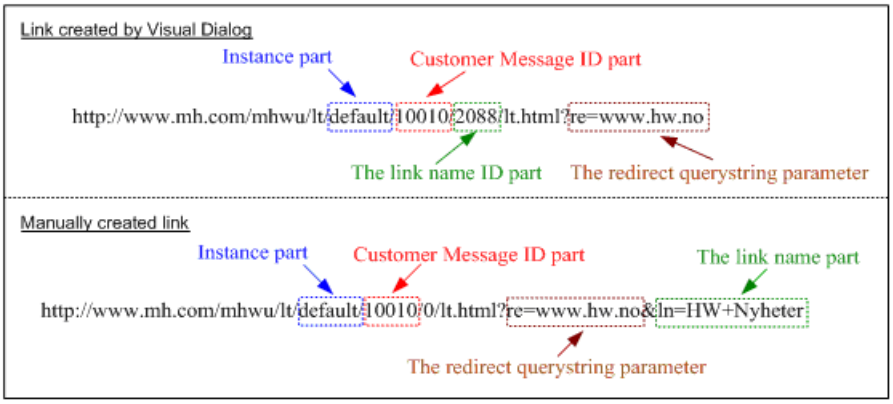
Die Verknüpfungsnachverfolgung identifiziert, auf welche Verknüpfung geklickt wurde, indem Informationen aus der URL extrahiert werden.

Die URL enthält die folgenden Parameter:

Parameter	Name
Instanz	Die Instanz. Erforderlich.
Kundennachricht-ID	Diese ID identifiziert den Kunden und die Nachricht. Dieser Wert entspricht der Datenbankspalte CUSTOMER_MESSAGE.CM_ID. Nicht erforderlich. Setzen Sie die Kundennachricht-ID auf „0“, wenn der Wert unbekannt ist, oder auf „nt“, um die Nachverfolgung zu deaktivieren.

Verknüpfungsname-ID	Die ID des Verknüpfungsnamens. Nur von Visual Dialogue verwendet. Nicht erforderlich. Setzen Sie den Wert auf „0“, wenn die Verknüpfung manuell erstellt wurde.
Verknüpfungsname	Der Name der Verknüpfung wird hauptsächlich für Berichtszwecke verwendet. Nicht erforderlich. Entfernen Sie diesen Teil, wenn er nicht benötigt wird.
Abfragezeichenfolgenparameter für die Umleitung	Die Ziel-URL. Erforderlich.
Prüfsummenparameter	Eine Prüfsumme, welche die Integrität der Ziel-URL sicherstellt. Konfigurierbar, ob sie erforderlich ist oder nicht. Wenn sie erforderlich ist, sind nur Umleitungen mit vorhandener und richtiger Prüfsumme zulässig. Wenn sie nicht erforderlich ist, sind auch Umleitungsverknüpfungen ohne eine Prüfsumme zulässig.

Die Verknüpfungsnachverfolgung unterstützt zwei verschiedene URL-Strukturen: eine, die von Visual Dialogue verwendet wird und eine, die verwendet wird, wenn die Verknüpfung außerhalb von Visual Dialogue erstellt wird. Der Unterschied ist, dass Visual Dialogue eine Verknüpfungsnamen-ID anstelle des Verknüpfungsnamens selbst verwendet. Da dies schwierig ist, wenn die Verknüpfung außerhalb von Visual Dialogue erstellt wird, ist es auch möglich, den Verknüpfungsnamen als Abfragezeichenfolgenparameter hinzuzufügen.



Hinweis: Wenn die Prüfsumme erforderlich ist, funktionieren manuell erstellte Verknüpfungen nicht, da sie keine Prüfsummen enthalten.

Aktivieren der Integritätsprüfung von Verknüpfungsnachverfolgungen (Prüfsumme)

Bei der Verknüpfungsnachverfolgung muss standardmäßig keine Prüfsumme in den Verknüpfungsnachverfolgungs-URLs vorhanden sein. Dadurch ist die Kompatibilität mit URLs in Nachrichten sichergestellt, die vor dem Upgrade auf Version 6.0.1 versendet wurden.

Um die Integritätsprüfung zu erzwingen (Prüfsumme erforderlich), muss eine Einstellung in der **web.config** unter <WebUtilities>\LT geändert werden:

```
<checkSum enable="true">
    <instances>
        <!--<instance name="InstanceName" enable="true" defaultRedirect=""
/>-->
    </instances>
</checkSum>
```

Die Konfigurationsdatei enthält auch Einstellungen für die Verwendung einer Umleitungs-URL, wenn die Prüfsumme falsch ist oder fehlt.

Es gibt einen Standardabschnitt in der Konfigurationsdatei, aber ggf. auch einen Datenabschnitt für jede einzelne Datenbankinstanz. Jeder Instanzabschnitt überschreibt die Standardwerte.

Hinweis: Es wird empfohlen, die Integritätsprüfung bei neuen PD-Installationen zu aktivieren.

E-Mail- und Verknüpfungsnachverfolgung – Details

Logging

Die Tabelle `WEB_TRACK_LOG` wird verwendet, um Informationen über nachverfolgte E-Mails und Verknüpfungen zu speichern. Die erste Tabelle unten zeigt die aufgezeichneten Informationen, wenn ein E-Mail-Empfänger ein Nachverfolgungsbild herunterlädt. Die Zweite zeigt die aufgezeichneten Informationen, wenn eine nachverfolgte Verknüpfung protokolliert wird.

Gespeicherte Informationen, wenn eine nachverfolgte E-Mail geöffnet wird

Gespeicherte Informationen	Verwendetes Feld in der Tabelle WEB_TACK_LOG
Datum und Zeitpunkt, zu dem das Bild heruntergeladen wurde	WTL_TIMESTAMP
Kundennachricht-ID	WTL_CM_ID
Protokollelementtyp (EMAIL_OPEN)	WTL_LOG_TYPE
Benutzeragentenzeichenfolge (eine Zeichenfolge, die den E-Mail-Browser identifiziert)	WTL_USER_AGENT
Browsertyp	WTL_BROWSER_TYPE
Browserversion	WTL_BROWSER_VERSION
Plattform (Betriebssystem)	WTL_PLATFORM

Informationen, die gespeichert werden, wenn auf eine nachverfolgte Verknüpfung geklickt wird

Gespeicherte Informationen	Verwendetes Feld in der Tabelle WEB_TACK_LOG

Datum und Zeitpunkt, zu dem das Bild heruntergeladen wurde	WTL_TIMESTAMP
Kundennachricht-ID **Hinweis:** Der Wert „0“ bedeutet, dass die Kundennachricht-ID (und der Kunde selbst) unidentifiziert sind.	WTL_CM_ID
Verknüpfungsname	WTL_LINK_NAME
Ziel-URL	WTL_URL
Protokollelementtyp (LINK)	WTL_LOG_TYPE
Benutzeragentenzeichenfolge (eine Zeichenfolge, die den E-Mail-Browser identifiziert)	WTL_USER_AGENT
Browsertyp	WTL_BROWSER_TYPE
Browserversion	WTL_BROWSER_VERSION
Plattform (Betriebssystem)	WTL_PLATFORM

Konfigurieren der Verknüpfungs- und E-Mail-Protokollierung

Die Datei „web.config“ der Anwendung „Web Utilities“ enthält Parameter dafür, wie Verknüpfungs- und E-Mail-Nachverfolgung gehandhabt werden.

Parametername	Beschreibung
LogMethod	Legt fest, welche Protokollierungsmethode verwendet wird. In der unteren Tabelle sind die verschiedenen unterstützten Protokollierungstypen aufgeführt.
SaveLogAfterXSeconds	Gibt die Anzahl der abzuwartenden Sekunden an, bevor das Protokollierungselement in der Datenbank/Datei gespeichert wird. Die Datensätze werden als Datei oder in der Datenbank gespeichert, wenn ein Protokollierungselement registriert wird und die Anzahl der Sekunden den angegebenen Wert überschreitet.
LogFileLocation	Speicherort der Protokollierungsdatei
MaxLogFileSize	Maximale Größe der Protokollierungsdatei in KB. Standardmäßig unbegrenzt.

Die E-Mail- und Verknüpfungsnachverfolgung unterstützt verschiedene Protokollierungsmethoden:

Methode	Beschreibung
Datenbank	Protokollierung nur in der Datenbank

Datei	Protokollierung nur als Datei
Datenbank und Datei	Protokollierung sowohl in der Datenbank als auch in der Datei
Datenbank oder Datei (Standard)	Protokollierung in der Datenbank. Protokollierung als Datei, wenn die Protokollierung in der Datenbank fehlschlägt.
Kein(e)	Keine Protokollierung.

Anwendungsstatus und Statistiken

Die Anwendung „Web Utilities“ beinhaltet eine Webseite für den einfachen Zugriff auf protokollierte Statistiken und Fehlermeldungen. Diese Informationen können angezeigt werden, indem die folgende URL aufgerufen wird:

http://<server>/<web_share>/Application/statistics.axd

Diese Seite ist standardmäßig deaktiviert. Sie kann aber aktiviert werden, indem im Anwendungsordner in der Datei „web.config“ der Parameter „StatisticsMode“ verändert wird.

Benutzerdefinierte Protokollierung

Benutzerdefinierte Anwendungen können einfach E-Mail- und Verknüpfungsnachverfolgung hinzufügen, indem die Webdienste von **WebUtilsAPI** verwendet werden.

Antwortnachverfolgung

Einleitung

Durch die Antwortennachverfolgung wird der Fortschritt in der Beantwortung der Fragen des Fragebogens durch den Befragten aufgezeichnet. Ein Protokollelement der Antwortnachverfolgung wird in der Datenbank für jede Fragebogenseite gespeichert, die ein Antwortender öffnet.

Verwenden der Antwortnachverfolgung

Visual Dialogue bietet einen Standardbericht für leichten Zugriff auf die Daten der Antwortnachverfolgung eines Fragebogens. Der Bericht zeigt die folgenden Informationen an:

- Die Anzahl der Befragten, die den Fragebogen vollständig ausgefüllt haben.
- Der Prozentsatz der Antwortenden, welche die Beantwortung einer der verschiedenen Seiten des Fragebogens abgeschlossen haben (im Vergleich zur Gesamtanzahl der Antwortenden, die mit der Beantwortung des Fragebogens begonnen haben).
- Den durchschnittlichen Zeitaufwand für die Beantwortung des Fragebogens.
- Die durchschnittliche Zeitdauer für die Beantwortung der verschiedenen Fragebogenseiten.

Anforderungen der Antwortnachverfolgung

Die Antwortnachverfolgung ist in den Modulen „Customer Web Access“ und „Telemarketing“ implementiert. In der Dokumentation von Visual Dialogue finden Sie Informationen zur Aktivierung der Antwortnachverfolgung.

Die Antwortnachverfolgung ist mit einem Fragebogenlayout verbunden, da das Layout eine Fragebogenseite definiert. Ein Fragebogen benötigt daher ein Layout, damit die Antwortnachverfolgung funktioniert.

Zeitpunkt der Aktivierung der Antwortnachverfolgung

Die Antwortnachverfolgung wird nur aktiviert, wenn der Antwortende den Fragebogen wirklich beantwortet, und nicht, wenn der Fragebogen getestet wird. Die Antwortnachverfolgung ist daher deaktiviert, wenn der Fragebogen in Visual Dialogue geöffnet wird (z. B. durch eine E-Mail im Nachrichtenmanager oder in der Nachrichtenvorschau). Die Antwortnachverfolgung kann ebenfalls deaktiviert werden, indem beim Erstellen einer Fragebogen-URL im Fragebogen-Designer „Antwortnachverfolgung deaktivieren“ ausgewählt wird.

Visual Dialogue deaktiviert die Antwortnachverfolgung, indem ein Abfragezeichenfolgeparameter, genannt „notrack“ zur URL hinzugefügt wird, die verwendet wird, um den Fragebogen zu öffnen. In E-Mails enthaltende Fragebogenverknüpfungen enthalten diesen Parameter, solange die E-Mail unter Verwendung von Visual Dialogue geöffnet wird. Er wird entfernt, wenn die E-Mail verschickt wird.

Protokollelemente der Antwortnachverfolgung

Die Antwortnachverfolgung verwendet drei verschiedene Typen von Protokollelementen, wenn eine Antwort nachverfolgt wird:

- Start – Das Protokollelement „Start“ wird registriert, wenn der Fragebogen das erste mal öffnet wird.
- Seitenaufruf – Das Protokollelement „Seitenaufruf“ wird registriert, wenn eine neue Seite öffnet wird.
- Abgeschlossen – Das Protokollelement „Abgeschlossen“ wird registriert, wenn die Beantwortung des Fragebogens abgeschlossen ist.

Zwischenspeichern

Zur Leistungssteigerung werden die Protokollelemente der Antwortnachverfolgung nicht direkt in der Datenbank gespeichert. Sie werden stattdessen auf dem Webserver zwischengespeichert und paketweise in der Datenbank abgespeichert. Die Protokollelemente werden standardmäßig in der Datenbank gespeichert, wenn ein neues Element registriert wird und mehr als 10 Sekunden seit dem letzten Speichern vergangen sind. Es ist jedoch möglich, die Zeit zwischen zwei Speichervorgängen zu ändern, indem für das Modul in der entsprechenden Datei „web.config“ die Einstellung „SaveResponseLogAfterXSeconds“ verwendet wird. Die zwischengespeicherten Protokollelemente der Antwortnachverfolgung werden auch gespeichert, wenn das Modul entladen wird.

Bei der Antwortnachverfolgung gespeicherte Informationen:

Die Tabelle QRY_RESPONSE_TRACK_LOG wird verwendet, um Informationen zur Antwortnachverfolgung zu speichern. Die Tabelle unten enthält die protokollierten Informationen:

Gespeicherte Informationen	Beschreibung	Verwendetes Feld in Tabelle QRY_RESPONSE_TRACK_LOG
Layout-ID	Die ID des Fragebogenlayouts beim Nachverfolgen von Antworten	QRTL_QL_ID
Antwortformular-ID	Wird nur festgelegt, wenn die Beantwortung des Fragebogens abgeschlossen ist.	QRTL_QAF_ID
Antwortkennung	Alle während einer einzelnen Antwort von einem Antwortenden registrierten Protokollelemente haben dieselbe Antwortkennung.	QRTL_RESPONSE_IDENTIFIER
Zeitstempel	Zeitpunkt der Registrierung des Protokollelements	QRTL_TIMESTAMP
Protokolltyp	Der Protokollelementtyp („Start“, „Seitenaufruf“ oder „Abgeschlossen“)	QRTL_LOG_TYPE
Seitenindex	Der Seitenindex des Fragebogens zum Zeitpunkt der Erstellung des Protokollelements	QRTL_PAGE_INDEX
Anonym	Bei einem anonym beantworteten Fragebogen	QRTL_IS_ANONYMOUS
Kundendomänen-ID	Die ID der Kundendomäne	QRTL_CD_ID
Kunden-ID	Die ID des Kunden	QRTL_CUSTOMER_ID
Kontext	Der Kontext	QRTL_CONTEXT
Dialogteilnehmer-ID	Die Dialogteilnehmer-ID	QRTL_DP_ID
Telemarketing-Projekt-ID	Die ID des Telemarketing-Projekts (wenn der Fragebogen durch das Telemarketing-Modul beantwortet wurde).	QRTL_CCP_ID
Kanaltypname	Der Name des Kanaltyps, der bei der Beantwortung verwendet wurde.	QRTL_CT_NAME
Übertragungs-ID	Die Übertragungs-ID. Übertragungen werden in der Datenbanktabelle FUZZY_MESSAGES gespeichert.	QRTL_FM_ID
URL	Die für den Zugriff auf den Fragebogen verwendete URL (wird nur für Protokollelemente des Typs „Start“ eingefügt).	QRTL_URL
Anmeldetyp	Der zur Beantwortung des Fragebogens verwendete Anmeldetyp (wird nur eingefügt, wenn der Fragebogen durch das Modul „Customer Web Access“ beantwortet wurde).	QRTL_LOGIN_TYPE
Aktualisierung des Antwortformulars	Wenn ein Protokollelement bei der Aktualisierung eines Antwortformulars registriert wird.	QRTL_IS_ANSWER_FORM_UPDATE

Benutzeragent	Die Zeichenfolge des Benutzeragenten des für die Beantwortung verwendeten Webbrowsers (wird nur bei Protokollelementen des Typs „Start“ eingefügt).	QRTL_USER_AGENT
Browsertyp	Der Browsertyp des für die Beantwortung des Fragebogens verwendeten Webbrowsers (wird nur bei Protokollelementen des Typs „Start“ eingefügt).	QRTL_BROWSER_TYPE
Browserversion	Die Browserversion des für die Beantwortung des Fragebogens verwendeten Webbrowsers (wird nur bei Protokollelementen des Typs „Start“ eingefügt).	QRTL_BROWSER_VERSION
Plattform	Die vom Antwortenden verwendete Plattform.	QRTL_PLATFORM

Kapitel 5

Inhaltsobjekte

In diesem Abschnitt:

- Inhaltsobjekte 100
- Inhaltsobjekt-URLs 100

Inhaltsobjekte

Inhaltsobjekte definieren dynamischen Inhalt. Die Platzierung des eigentlichen Inhalts wird durch die Auswertung eines Satzes von Regeln entschieden. Sie werden normalerweise in Fragebögen, E-Mails und für Profilformate verwendet. Sie können aber auch für beliebige Webseiten verwendet werden, solange die Webanwendung „Web Utilities“ verfügbar ist.

Inhaltsobjekt-URLs

Inhaltsobjektmodi

Inhaltsobjekte können unter Verwendung von einem von zwei Modi angezeigt werden: Seitenmodus oder eingebundener Modus.

Seitenmodus

Ein im Seitenmodus angezeigtes Inhaltsobjekt wird mittels einer ganzen HTML-Seite angezeigt und muss unter Verwendung eines iFrame oder ähnlichem angezeigt werden.

Das folgende Beispiel zeigt, wie ein Inhaltsobjekt in einen iFrame eingebunden werden kann:

<iframe src="http://server/mhwu/CO/ContentObject.axd?

om=page&cok=Content_1" ></iframe>

Eingebundener Modus

Ein im eingebundenen Modus angezeigtes Inhaltsobjekt wird durch einen Teil-HTML-Code angezeigt und muss in eine andere Webseite eingebunden werden. Das Inhaltsobjekt wird der Webseite hinzugefügt, indem ein Skripttag mit einem Verweis zum Inhaltsobjekt an der Stelle erstellt wird, wo es angezeigt werden soll.

Das folgende Beispiel zeigt, wie ein Inhaltsobjekt in eine Webseite eingebunden werden kann:

<script src="http://dev004/mhwu/co/Object.axd?

om=wrapped&cok=CONTENT_1000&customerid=106981&showerror" type="text/javascript"></script>

Inhaltsobjekt-URL-Struktur

Die URL, die ein Inhaltsobjekt angibt, kann die folgenden Parameter enthalten:

Parameter	Parameterschlüssel	Name

Inhaltsobjektmodus	OM	Anzeigeart des Inhaltsobjekts (erforderlicher Parameter). Gültige Werte: • „Page“ – Das Inhaltsobjekt wird als volle HTML-Seite angezeigt. • „Wrapped“ – Das Inhaltsobjekt wird als Teil-HTML-Code angezeigt.
Inhaltsobjektschlüssel	COK	Inhaltsobjektschlüssel (erforderlicher Parameter)
Instanz	Instanz	Legt fest, zu welcher Instanz von Dialogue Server das Inhaltsobjekt gehört. Ohne Angabe wird die Standardinstanz verwendet.
Domänen-ID	CustDomainId	Kundendomänen-ID
Kunden-ID	CustomerId	Die Kunden-ID kann angegeben werden, um kundenspezifische Inhalte anzuzeigen.
Kontext	Kontext	Der Kontext. Standardwert: „0“.
Teilnehmer-ID	ParticipantId	Die Teilnehmer-ID kann anstelle der Kunden-ID verwendet werden, um kundenspezifische Inhalte anzuzeigen.
Fehler anzeigen	showError	Gibt an, ob eine Fehlermeldung angezeigt werden soll, wenn ein Fehler bei der Anzeige des Inhaltsobjekts auftritt. Dieser Parameter wird hauptsächlich für Debugging-Zwecke verwendet.
Ursprung	Ursprung	Ein optionaler Wert, der den Ursprung der Inhaltsobjektanfrage angibt. Dieser Wert wird in einer getrennten Spalte (col_object_log) in der Datenbanktabelle (content_object_log) gespeichert, wo Inhaltsobjektausführungen protokolliert werden. Daher ist der Parameter für Berichte geeignet.
Keine Nachverfolgung	notrack	Falls die Verknüpfungsnachverfolgung für das Inhaltsobjekt verwendet wird, kann sie durch diesen Parameter deaktiviert werden.
Zusätzliche Parameter		Sie können beliebige zusätzliche Parameter einfügen. Diese Parameter können in den Inhaltselementen und -regeln eines Inhaltsobjekts abgerufen und verwendet werden.
Zwischenspeichern	C	Ob und wie der Cache verwendet wird. Durch Angabe des Parameters wird der Cache nicht verwendet. Dies ist auch das Standardverhalten, wenn kein Parameter „C“ angegeben wird. **Use** – Das Inhaltsobjekt wird aus dem Cache zurückgegeben, wenn es dort gefunden wird und vom Dialogue Server kommt. **Clear** – Entfernt das Inhaltsobjekt aus dem Cache und gibt eine frische Kopie aus Dialogue Server zurück. **ClearOnly** – Entfernt das Inhaltsobjekt aus dem Cache, ohne etwas zurückzugeben.

		Hinweis: Inhaltsobjekte werden basierend auf den folgenden Parametern zwischengespeichert: Ohne Angabe einer Teilnehmer-ID: Inhaltsobjektmodus, Inhaltsobjektschlüssel, Instanz, Domänen-ID, Kunden-ID, Kontext und zusätzliche Parameter. Bei Angabe einer Teilnehmer-ID: Inhaltsobjektmodus, Inhaltsobjektschlüssel, Instanz, Teilnehmer-ID und zusätzliche Parameter.
Zeitüberschreitung der Zwischenspeicherung	CTO	Die Zeitdauer, in der das Inhaltsobjekt zwischengespeichert wird (in Minuten). Die Zeitüberschreitung ist gleitend. Das bedeutet, sie wird bei jedem Zugriff auf das Element zurückgesetzt. Der Standardwert ist 5 Minuten.

Die untere Abbildung zeigt die URL-Struktur des Inhaltsobjekts:

Abbildung 1 – Struktur der Inhaltsobjekt-URL

Kapitel 6

Veröffentlichte Dateien

In diesem Abschnitt:

- Veröffentlichte Dateien 104
- URLs veröffentlichter Dateien 104

Veröffentlichte Dateien

Eine veröffentlichte Datei ist eine auf dem Dialogue Server gespeicherte und dort verfügbare Datei. Sie ist im Internet durch speziell erstellte URLs aufrufbar und kann auf beliebigen Webseiten verwendet werden.

URLs veröffentlichter Dateien

URL-Struktur von veröffentlichten Dateien

Die eine veröffentlichte Datei angebende URL enthält die folgenden Parameter:

Parameter	Parametername	Name
Instanz		Legt fest, zu welcher Instanz von Dialogue Server die veröffentlichte Datei gehört.
Inhalts-ID		Die Inhalts-ID der veröffentlichten Datei.
Dateiname		Gibt den Dateinamen der veröffentlichten Datei an.
Aktualisierungsmodus (optional)	refreshMode	Gibt an, wie mit zwischengespeicherten Versionen der Datei verfahren wird. Die untere Abbildung zeigt ein Beispiel. **Zulässige Parameterwerte:** • ***0*** – Datei zurückgeben. Gibt die Datei aus dem Cache zurück, falls verfügbar, ansonsten wird sie von Dialogue Server abgerufen. • ***1*** – Cache löschen. Entfernt alle zwischengespeicherten Versionen der Datei (gibt die Datei nicht zurück). • ***2*** – Cache löschen und zurückgeben. Löscht alle zwischengespeicherten Versionen der Datei, und ruft eine aktuelle Version aus Dialogue Server ab.

Dialog „Speichern unter...“ verwenden (optional)	useSaveAsDialog	Der Dialog „Speichern unter...“ wird abhängig vom Dateityp der veröffentlichten Datei angezeigt. Das Standardverhalten kann durch Angabe dieses Parameters überschrieben werden. Standardmäßig wird der Dialog „Speichern unter...“ für alle Dateitypen außer den folgenden angezeigt: • ***Bilder*** – Dateien mit den Erweiterungen bmp, gif, jpg, jpeg und png. • ***HTML-Dateien*** – Dateien mit den Erweiterungen htm und html. • ***PDF-Dateien*** **Zulässige Parameterwerte:** • ***true*** – Der Dialog „Speichern unter...“ wird verwendet. • ***false*** – Der Dialog „Speichern unter“ wird nicht verwendet.

Die Abbildung unten zeigt die URL-Struktur einer veröffentlichten Datei:

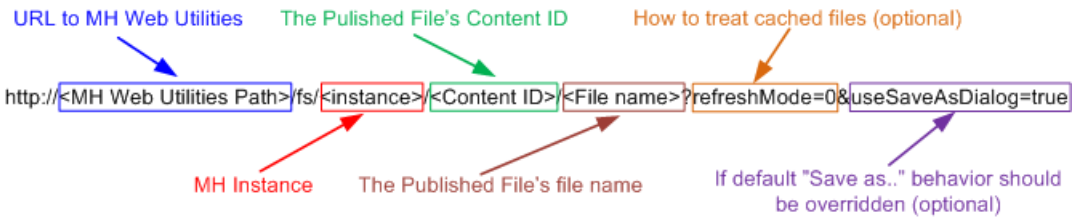


Abbildung 1 – URL-Struktur einer veröffentlichten Datei

Kapitel

7

Verwaltung unzustellbarer E-Mails

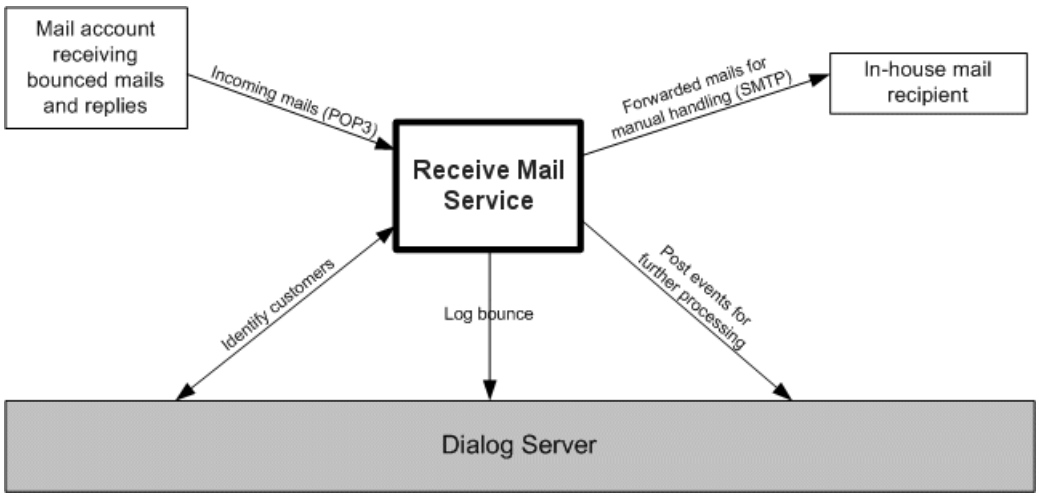
In diesem Abschnitt:

- Verwaltung unzustellbarer E-Mails 108
- Systemparameter unzustellbarer E-Mails 108
- Protokollierung unzustellbarer E-Mails 109

Verwaltung unzustellbarer E-Mails

Verwaltung unzustellbarer E-Mails – Einführung

Unzustellbare E-Mails werden in Dialogue Server vom *Nachrichtenempfangsdienst* verwaltet. Dieser Dienst ruft E-Mails von einem E-Mail-Konto auf einem Server ab, der unzustellbare E-Mails empfängt. Die E-Mails werden entsprechend der Einstellung **Einrichtung für unzustellbare E-Mails** in Dialogue Admin verwaltet. Die unzustellbaren E-Mails können dann unter Verwendung der Operation *Unzustellbare E-Mail überprüfen* weiter verarbeitet werden.



Der E-Mail-Empfangsdienst ist abhängig von einem Drittanbieterprodukt, BoogieBounce, um die verschiedenen Typen der Unzustellbarkeit zu identifizieren. Sie können eine Lizenz für dieses Produkt durch den Kundensupport erwerben.

Systemparameter unzustellbarer E-Mails

Systemparameter

Der *Nachrichtenempfangsdienst* kann in Dialogue Admin unter **Parametersammlung** > ***Nachrichtenempfangsdienst*** konfiguriert werden. Die folgenden Parameter müssen festgelegt werden:

Parametername	Beschreibung
BBLicenseKey	Der Lizenzschlüssel für die API „BoogieBounce“. Dies ist der Lizenzschlüssel, den Sie beim Kauf der API „BoogieBounce“ über den Kundensupport erhalten.

CheckMailInterval	Dies ist das Intervall in Sekunden, das angibt, wie häufig der Nachrichtenempfangsdienst eingehende E-Mails auf Adressen prüft, denen keine E-Mail zugestellt werden kann.
EnableService	Aktivierung oder Deaktivierung des Nachrichtenempfangsdienstes für die aktuelle Instanz. Dieser Parameter muss auf TRUE gesetzt werden, damit der Nachrichtenempfangsdienst eingehende E-Mails überprüft. Dies überschreibt die Option *Konfiguration aktiviert* einer einzelnen **Unzustellbarkeitskonfiguration**.
WorkingDirectory	Das interne Arbeitsverzeichnis, das vom Nachrichtenempfangsdienst verwendet wird. In dieser Ordnerstruktur werden alle unzustellbaren E-Mails als eine separate Datei pro Unzustellbarkeitscode gespeichert.

Protokollierung unzustellbarer E-Mails

Logging

Informationen zu unzustellbaren E-Mails werden in der Tabelle BOUNCE_EMAIL_LOG gespeichert. Welche Informationen gespeichert werden, ist davon abhängig, wie viele Informationen aus der unzustellbaren E-Mail vom Dienst extrahiert werden konnten. Zusätzlich wird die Tabelle MESSAGE_LOG aktualisiert, wenn ein Ereignis für die unzustellbare E-Mail generiert wird.

In Tabelle BOUNCE_EMAIL_LOG gespeicherte Informationen

Gespeicherte Informationen	Verwendetes Feld in Tabelle BOUNCE_EMAIL_LOG
Kundendomänen-ID	BEL_CD_ID
Kundennachricht-ID	BEL_CM_ID
Nachrichtenpaket-ID	BEL_MBL_ID
Kunden-ID	BEL_CUSTOMER_ID
Kontext	BEL_CONTEXT
Beschreibt, wie die ursprüngliche E-Mail gefunden wurde.	BEL_IDENTIFIED_BY
Unzustellbarkeitscode	BEL_BOUNCE_CODE
Die in der unzustellbaren E-Mail gefundene E-Mail-Adresse.	BEL_EMAIL_ADDRESS
Der Ort, an dem sich die ursprüngliche unzustellbare E-Mail befindet.	BEL_UNC

Der Zeitpunkt, zu dem das Unzustellbarkeitsprotokoll durch den Nachrichtenempfangsdienst erstellt wurde.	BEL_TREATED_TIMESTAMP
Der Zeitpunkt, zu dem der Nachrichtenempfangsdienst ein Ereignis für die unzustellbare E-Mail erstellt hat.	BEL_SYSTEM_PROCESSED_TIMESTAMP
Dieses Feld wird von benutzerdefinierten Anwendungen zur Protokollverarbeitung verwendet.	BEL_CUSTOM_PROCESSED_TIMESTAMP

In Tabelle MESSAGE_LOG gespeicherte Informationen

Gespeicherte Informationen	Verwendetes Feld in Tabelle MESSAGE_LOG
Boolesches Feld, das auf „T“ gesetzt wird, wenn die Nachricht unzustellbar ist.	ML_BOUNCED
Unzustellbarkeitscode	ML_BOUNCE_CODE
Der Zeitpunkt, zu dem die unzustellbare Nachricht vom Nachrichtenempfangsdienst verarbeitet wurde.	ML_BOUNCED_TIMESTAMP

Kapitel

8

Importieren und Exportieren von Objekten

In diesem Abschnitt:

- Exportieren und Importieren 112
- Exportieren 112
- Importieren 113
- Einrichten einer komplementären Umgebung 114

Exportieren und Importieren

Exportieren und Importieren – Einführung

Portrait Dialogue bietet Ihnen die Möglichkeit, die meisten seiner Objekte zu exportieren und zu importieren. Dies bedeutet, dass Sie ein Objekt in einem System entwickeln, es exportieren und in ein anderes System importieren können. Objekte können abhängig vom Objekttyp sowohl in Visual Dialogue als auch in Dialogue Admin exportiert und importiert werden.

Komplementäres und nicht-komplementäres Importieren

Das Importieren kann in zwei verschiedenen Modi durchgeführt werden: *komplementär* und *nicht-komplementär*.

Komplementäres Importieren wird verwendet, wenn zwei oder mehr Instanzen besonders konfiguriert wurden, um nahtloses Exportieren und Importieren zwischen den Instanzen bereitzustellen. Dies wird typischerweise verwendet, wenn Sie eine Entwicklungs-/Produktionsumgebung benutzen, in der Sie in einem oder mehreren Systemen entwickeln und testen, und dann Ihre Implementierung in ein Produktionssystem verschieben möchten.

Nicht-komplementäres Importieren wird zwischen zwei beliebigen Systemen verwendet, aber kann weitere Interaktionen durch den Benutzer erfordern, damit während und nach dem Importieren alles funktioniert. Dies kann beispielsweise verwendet werden, um einen Dialog zum Kundensupport zu senden, mit dem Sie Probleme haben.

Exportieren

Exportieren aus Visual Dialogue

Sie können aus Visual Dialogue Folgendes exportieren:

- Dialoge (einschließlich Telemarketing-Projekte)
- selections
- Nachrichtenvorlagen
- Fragebögen
- Berichtsvorlagen
- Übertragung
- Fragebogenformate
- Veröffentlichte Dateien
- Inhaltsobjekte

Alle Importvorgänge werden über einen Assistenten durchgeführt, der Sie durch den Vorgang führt. Weitere Informationen finden Sie in der Dokumentation von Visual Dialogue.

Exportieren aus Dialogue Admin

Sie können aus Dialogue Admin Folgendes exportieren:

- Kundendomänen
- Gruppentypen
- Operationstypen
- SQL-Definitionen
- Plug-In-Definitionen

Wählen Sie zum Durchführen die Option *Exportieren...* aus dem Kontextmenü oder aus dem Hauptmenü *Element* aus.

Exportieren von Kundendomänen

Wenn Sie eine Kundendomäne exportieren, werden alle SQL-Abfragen und die verwendeten Plug-Ins automatisch mit exportiert. Für die Kundendomäne definierte Kategorien werden ebenfalls mit eingeschlossen.

Hinweis: Wenn Sie eine Kundendomäne importieren, werden Kategorien nicht aktualisiert oder gelöscht, es werden nur neue Kategorien eingefügt. Dies verhindert den unbeabsichtigten Verlust von Kundendaten.

Exportieren von Operationstypen

Wenn Sie einen Operationstypen exportieren, werden die von den Verzweigungen in den Operationen verwendeten Plug-Ins automatisch mit exportiert.

Importieren

Allgemeines Importieren

Importvorgänge werden sowohl in Visual Dialogue als auch in Dialogue Admin über einen Importassistenten durchgeführt. Dieser Assistent wird ausführlich in der Dokumentation von Visual Dialogue beschrieben. Sie können den Assistenten in Dialogue Admin entweder im Menü „Tools“ aufrufen, oder indem Sie auf ein Element in der Baumansicht mit der rechten Maustaste klicken und *Importieren...* auswählen.

Nicht-komplementäres Importieren

Nicht-komplementäres Importieren wird zwischen zwei beliebigen Systemen durchgeführt. Keines der Objekte, das Sie importieren, darf im Zielsystem existieren. Der Importassistent bietet die Möglichkeit, zu importierende Objekte umzubenennen. Daher werden alle importierten Objekte neue Objekte im Zielsystem sein. Es ist nicht möglich, bestehende Objekte zu überschreiben oder zu aktualisieren.

Viele Objekte sind abhängig von anderen Objekten oder Systemdaten. Dies kann zu Problemen beim Importieren führen. Der Importassistent zeigt diese Abhängigkeiten an und bietet unter gewissen Umständen die Möglichkeit, diese zu ändern oder zu löschen.

Komplementäres Importieren

Komplementäres Importieren wird verwendet, wenn zwei oder mehr Systeme besonders konfiguriert wurden, um nahtloses Exportieren und Importieren zwischen den Systemen bereitzustellen. Dies wird typischerweise verwendet, wenn Sie eine Entwicklungs-/Produktionsumgebung benutzen, in der Sie in einem oder mehreren Systemen entwickeln und testen, und dann Ihre Implementierung in ein Produktionssystem verschieben möchten.

Wenn Sie in einer so konfigurierten Umgebung arbeiten, ist es wichtig, alle Änderungen in einem System auszuführen, und dann in die anderen Systeme zu exportieren, um die Umgebung synchronisiert zu halten. Sie sollten beispielsweise keinen neuen Dialog mit demselben Namen in zwei verschiedenen Systemen erstellen. Dies wird während des Importvorgangs als Synchronisierungsproblem erkannt und verhindert das Abschließen des Vorgangs.

Objekte, die nicht exportiert oder importiert werden können (typischerweise in Dialogeinrichtungen und anderen Typen in Dialogue Admin), müssen manuell synchronisiert werden. Zu diesem Zweck werden sie in allen Systemen mit identischen technischen Namen erstellt.

Es gibt keine Optionen zum Beheben von Problemen mit Abhängigkeiten, wenn Sie komplementär importieren. Der Grund dafür ist, dass komplementäre Systeme entweder durch Exportieren/Importieren oder durch die manuelle Erstellung bestimmter Objekte synchronisiert sein sollen. Alle benötigten Objekte müssen entweder Teil des Importvorgangs selbst oder bereits im Zielsystem definiert sein.

Komplementäres Importieren von Dialogen und Fragebögen kann zu Datenverlust führen. Wenn Sie einen Dialog dort importieren, wo eine Gruppe gelöscht wurde, wird diese Gruppe inklusive Teilnehmer ebenfalls im Zielsystem gelöscht. Dasselbe gilt für Fragen und Alternativen in Fragebögen.

Dies bedeutet: Wenn Sie eine Frage löschen, und dann mit demselben Namen und Frageschlüssel erneut erstellen, bleiben die ursprüngliche Frage und ihre Antworten immer noch gelöscht, wenn Sie einen Importvorgang durchführen. Der Grund dafür ist, dass die Identifizierung von Gruppen, Operationen, Verzweigungen, Fragen und Alternativen anhand interner Kennungen durchgeführt wird, und nicht durch den Frageschlüssel.

Einrichten einer komplementären Umgebung

Eine komplementäre Umgebung basiert auf der Annahme, dass alle exportierten/importierbaren Objekte denselben Kennungswert in allen Systemen besitzen. Objekte desselben Typs mit derselben Kennung werden in den Systemen als dasselbe Objekt betrachtet. Wenn Sie eine komplementäre Umgebung erstellen, müssen Sie zwei Dinge durchführen, um dies zu erreichen.

1) Erstellen identischer Systeme

Sie beginnen mit der Einrichtung der Umgebung, indem Sie eine bestehende Systemdatenbank (für gewöhnlich die Datenbank Ihres Produktionssystems) als Basisdatenbank einrichten. Anschließend erstellen Sie eine oder mehrere Kopien dieser Datenbank und verwenden Sie als Systemdatenbanken für die Entwicklung, Tests oder Vorproduktion. Nachdem Sie dies durchgeführt haben, besitzen Sie zwei oder mehrere identische Systeme. Wie Sie eine Kopie der Datenbank erstellen ist abhängig vom verwendeten RDBMS-Typ.

2) Konfigurieren der Systemparameter

Es gibt vier Systemparameter, die festgelegt werden müssen, damit die komplementäre Umgebung eingerichtet ist und funktioniert. Sie müssen sofort nach der Erstellung der identischen Systeme festgelegt werden, bevor etwas anderes mit den Systemen durchgeführt wird. Alle Parameter müssen für jedes System festgelegt werden. Sie befinden sich in Dialogue Admin in der Parametersammlung *Systemumgebung*.

- **ComplementaryEnvironmentName**

  Dies ist der Name der komplementären Umgebung. Dieser Wert muss derselbe in allen Systemen sein, da Sie komplementäres Importieren nur zwischen Systemen ausführen können, die denselben Umgebungsnamen haben.

- **ComplementarySystemName**

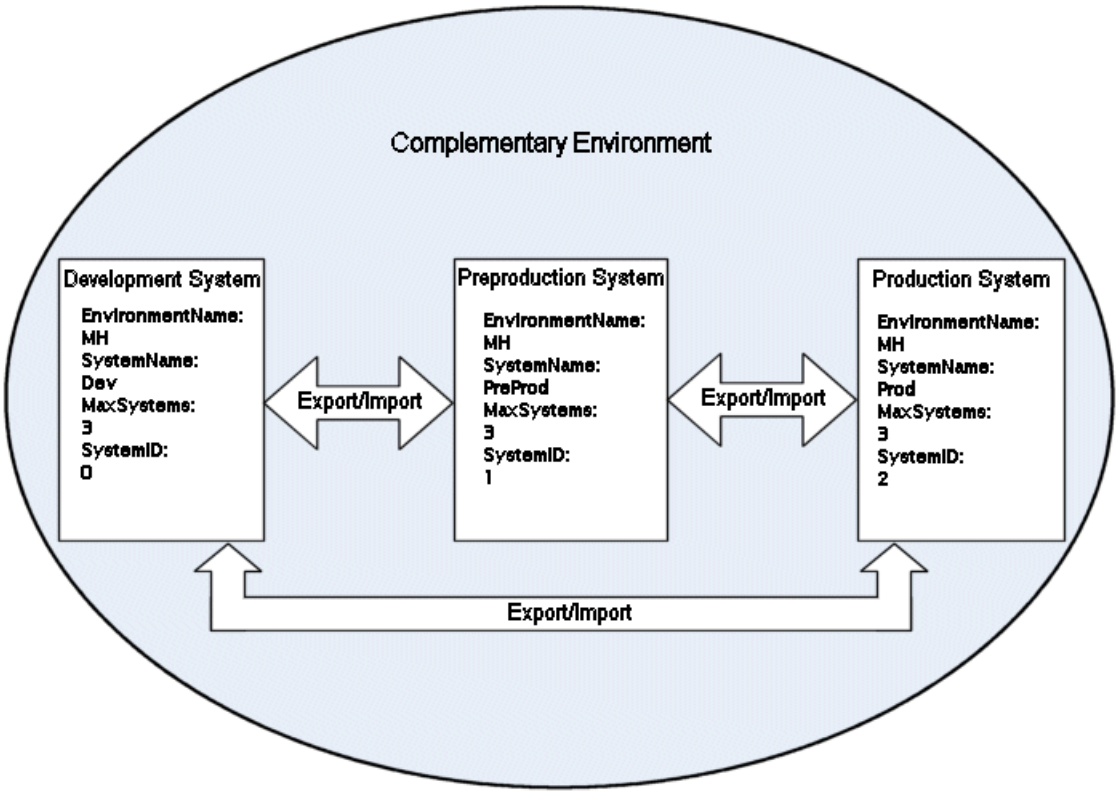
  Dies ist ein eindeutiger Name für jedes System in der komplementären Umgebung. Der Wert muss daher für jedes System unterschiedlich sein.

- **ComplementaryMaxSystems**

  Die maximale Anzahl der Systeme in der komplementären Umgebung. Wenn Sie eine Entwicklungs-, ein Vorproduktions- und ein Produktionssystem haben, muss dieser Wert in allen Systemen „3“ sein. Sie können eine größere Zahl angeben, damit weitere Systeme später hinzugefügt werden können, jedoch nicht weniger als die anfängliche Anzahl. Wenn Sie zu einem späteren Zeitpunkt die Umgebung ändern möchten, um mehr Systeme als der Wert dieses Parameters zu haben, muss der gesamte Prozess zur Erstellung identischer Systeme erneut durchgeführt werden. Daher ist es sinnvoll vorauszuplanen.

- **ComplementaySystemID**

  Dies ist eine eindeutige Kennung in jedem komplementären System. Der Wert sollte zwischen Null und einem weniger als die maximale Anzahl der komplementären Systeme liegen (festgelegt im Parameter *ComplementaryMaxSystems*). Dieser Wert muss für alle Systeme der Umgebung unterschiedlich sein. Ein Setzen auf „-1“ bewirkt, dass das System nicht Teil der komplementären Umgebung ist.


Complementary Environment
Development System
EnvironmentName:
MH
SystemName:
Dev
MaxSystems:
3
SystemID:
0
Export/Import
Preproduction System
EnvironmentName:
MH
SystemName:
PreProd
MaxSystems:
3
SystemID:
1
Export/Import
Production System
EnvironmentName:
MH
SystemName:
Prod
MaxSystems:
3
SystemID:
2
Export/Import

9

URL shortening

In this section:

- URL-Kürzung 118
- Datenbankinstanzen 119
- Parameter 119
- Nachrichtentypen 120

URL-Kürzung

Die **URL-Kürzung** ist ein Mechanismus zur Erstellung von Aliasen für URLs. Mit ihr können kurze URL-Aliase für oft lange URLs erstellt werden.

Eine **Fragebogen-URL** in Portrait Dialogue ist beispielsweise eine sehr lange URL, die man in einer SMS-Nachricht schwer versenden kann. Diese URL kann vor dem Versenden der SMS-Nachricht mithilfe des Mechanismus gekürzt werden:

- **Ursprüngliche Fragebogen-URL:**

```
http://www.pb.com/MHCwa/DefaultAn.aspx?Instance=5D085108400047&LoginTy-
pe=0C&QuestionnaireID=085C055F&LayoutID=085C065B&BroadcastID=&CheckS-
um=4C5C065002
```

- **Gekürzte URL:**

```
http://shorturl.pb.com/AhF56yx
```

In Portrait Dialogue werden zwei verschiedene Arten von URL-Kürzungen unterstützt.

1. **Codierte gekürzte URL:**

   Dabei handelt es sich um ein sehr kurzes kryptisches Wort, den sogenannten **gekürzten URL-Schlüssel**. Eine codierte gekürzte URL kann für jede Art von Hyperlink verwendet werden, der von Portrait Dialogue (entweder identifiziert oder anonym) erstellt oder aus dem Internet kopiert wurde. Sie ist mehr für digitale Dokumente geeignet, bei denen sich der Link anklicken lässt.

   Beispiel: `http://www.pb.com/mhwu/su/AhF56yx`

2. **Benannte gekürzte URL:**

   Dabei handelt es sich um ein einzelnes, vom Benutzer festgelegtes Wort, das aussagekräftiger sein kann, z. B. `Weihnachten2013`. Eine benannte gekürzte URL kann für verschiedene Hyperlinke verwendet werden, die von Portrait Dialogue (nur anonym) erstellt oder aus dem Internet kopiert wurden. Dieser URL-Typ eignet sich eher für Links in gedruckten Dokumenten. NB! Die Datenbanktabelle der gekürzten URLs ist für alle Kundendomänen der Instanz dieselbe. Eine benannte gekürzte URL darf nur für eine Instanz verwendet werden.

   Beispiel: `http://www.pb.com/mhwu/su/jw/Weihnachten2013`

Für die Erstellung geeigneter und noch kürzerer URLs für beide Arten von gekürzten URLs konfigurieren Sie IIS am besten so, dass eine Weiterleitung zum Webdienstprogramm für gekürzte URLs erfolgt, wie in den Parametern **Öffentliche URLs** von Dialogue Admin definiert. Das heißt:

- **Codierte gekürzte URL:**

```
http://shorturl.pb.com/AhF56yx
```

- **Benannte gekürzte URL:**

```
http://shorturl.pb.com/Christmas2013
```

Die URL-Kürzung steht im Nachrichtendesigner von **Visual Dialogue** und im **Web-Nachrichten-Designer** beim Einfügen von Hyperlinks zur Verfügung sowie über **Ausdrucksfunktionen** und die **Web Utilities-API**.

Datenbankinstanzen

Eine **Datenbankinstanz** besitzt eine Kurzkennung als Eigenschaft. Mithilfe der Kurzkennung können Instanzen ohne Probleme für die gekürzten URLs zwischen Servern verschoben werden. Bei einer Site mit mehr als einer Datenbankinstanz ist es wichtig, jeder Instanz eine Kurzkennung zuzuweisen.

Bei einer **codierten gekürzten URL** wird die Kurzkennung verschlüsselt in den**gekürzten URL-Schlüssel** integriert.

Beispiel: `http://www.pb.com/mhwu/su/AhF56yx`

Bei einer **benannten gekürzten URL** muss die Kurzkennung hingegen in der von der Web Utilities-Anwendung angeforderten URL enthalten sein.

Beispiel: `http://www.pb.com/mhwu/su/jw/Weihnachten2013`

Parameter

Die URL-Kürzungsfunktion verwendet beim Erstellen gekürzter URLs **Parameterwerte** von Dialogue Admin. Diese Parameter sind unter den Parametern **Öffentliche URLs** in Dialogue Admin zu finden.

Parameter WebUtilsURL

Zur Erstellung gekürzter URLs verwendet Dialogue Server standardmäßig die Basis-URL für die Web Utilities-Anwendung, die über den Parameter **WebUtilsURL** konfiguriert wird. Beispiele:

- **Codierte gekürzte URL:**

  ```
  http://www.pb.com/mhwu/su/AhF56yx
  ```

- **Benannte gekürzte URL:**

  ```
  http://www.pb.com/mhwu/su/jw/Christmas2013
  ```

Für die Erstellung geeigneter und noch kürzerer URLs können Systemadministratoren Weiterleitungen in IIS festlegen. Der Dialogue Server muss bei der Erstellung gekürzter URLs über solche Überschreibungen „informiert“ sein. Dafür müssen weitere Parameter in Dialogue Admin konfiguriert werden.

Parameter BaseShortenedURL

Der Parameter **BaseShortenedURL** überschreibt **WebUtilsURL** als Basis für die Erstellung gekürzter URLs. Beispiele:

- **BaseShortenedURLParameterwert**

  ```
  http://shorturl.pb.com
  ```

  Konfigurierte Weiterleitung

  ```
  http://shorturl.pb.com -> http://www.pb.com/mhwu/su
  ```

  Resultierende codierte gekürzte URL:

  ```
  http://shorturl.pb.com/AhF56yx
  ```

  Resultierende benannte gekürzte URL:

  ```
  http://shorturl.pb.com/jw/Christmas2013
  ```

Parameter NamedShortenedURL

Bei benannten gekürzten URLs ist es möglich, den Parameter **NamedShortenedURL** anzugeben, um den Instanzteil (`jw` in den Beispielen) in den gekürzten URLs wegzulassen. Beispiele:

- **NamedShortenedURL Parameterwert**

  ```
  http://shorturl.pb.com
  ```

  Konfigurierte Weiterleitung

  ```
  http://shorturl.pb.com -> http://www.pb.com/mhwu/su/jw
  ```

  Resultierende benannte gekürzte URL:

  ```
  http://shorturl.pb.com/Christmas2013
  ```

- **NamedShortenedURL Parameterwert**

  ```
  http://shorturl.pb.com/Norway
  ```

  Konfigurierte Weiterleitung

  ```
  http://shorturl.pb.com/Norway -> http://www.pb.com/mhwu/su/jw
  ```

  Resultierende benannte gekürzte URL:

  ```
  http://shorturl.pb.com/Norway/Christmas2013
  ```

Nachrichtentypen

Auf der Registerkarte **Design** in **Nachrichtentypen** gibt es ein Kontrollkästchen namens

URL-Kürzung standardmäßig im Designer verwenden.

Wird dieses Kontrollkästchen aktiviert, wird beim Einfügen eines Weblinks oder eines Hyperlinks in eine Nachrichtenvorlage in Portrait Dialogue eine Maske mit den Hyperlinkeinstellungen angezeigt. Mithilfe dieser Maske können Sie Eigenschaften wie die Aktivierung/Deaktivierung der URL-Kürzung oder der Verknüpfungsnachverfolgung festlegen.

Die Option zur Anzeige der gekürzten URL-Form ist standardmäßig nur für SMS und Text-E-Mails aktiviert.

Kapitel 10

Plug-In-API

In diesem Abschnitt:

- Plug-In-API 122
- Kundendaten-Plug-Ins 123
- Verzweigungs-Plug-Ins 126
- Ausdrucks-Plug-Ins 127
- Generische Plug-Ins 128
- Nachrichten-Plug-Ins 128
- Ausgangskanal-Plug-Ins 129
- Typen 131
- Schnittstellen 138

Plug-In-API

Einführung in Plug-Ins

Das Konzept von Plug-Ins ist fundamental für Portrait Dialogue. Plug-Ins werden in Dialogue Server verwendet, um Funktionen zu implementieren und anzupassen.

Zum Beispiel kann die Logik hinter einer Operation in einem Dialog durch das Schreiben eines Plug-Ins implementiert werden.

Ein Plug-In ist ein Stück Software oder Code, das nach definierten Richtlinien geschrieben wurde.

Plug-Ins werden als *COM-Objekte*, .NET-Klassen oder Skripte implementiert und installiert. Der Skriptansatz verwendet **Windows Script** und unterstützt installierte Skriptsprachen wie VBScript, JScript und PerlScript. Dialogue Server lädt und ruft die Plug-Ins auf, wenn sie benötigt werden, z. B. beim Ausführen einer Operation.

Verschiedene Arten von Plug-Ins

Das Verhalten von Dialogue Server kann an verschiedenen Stellen angepasst werden. Daher werden Plug-Ins für verschiedene Zwecke verwendet. Es gibt sechs Gruppen von Plug-Ins:

- **Kundendaten-Plug-Ins**

  Diese Plug-Ins werden in der Definition von Kundendomänen verwendet, um Kundendaten zu aktualisieren.

- **Plug-Ins für Dialogverzweigungen**

  Diese Plug-Ins implementieren Verzweigungen in Dialogoperationen**.** Viele der Standardverzweigungs-Plug-Ins, die mit dem System installiert werden, sind implementiert als Open-Source-Skripte.

- **Ausdrucks-Plug-Ins**

  Diese Plug-Ins implementieren benutzerdefinierte Ausdrucksfunktionen.

- **Nachrichten-Plug-Ins**

  Diese Plug-Ins sind zuständig für das Erstellen von Nachrichten. Eines der Standard-Plug-Ins fügt beispielweise Word-Vorlagen mit Kundendaten zusammen.

- **Ausgangskanal-Plug-Ins**

  Diese Plug-Ins implementieren die physische (ausgehende) Kommunikation durch einen Kanal. Beispielsweise sendet das Plug-In „E-Mail senden“ E-Mail-Nachrichten mithilfe des SMTP-Protokolls.

- **Generische Plug-Ins**

  Diese Plug-Ins implementieren benutzerdefinierte Logik.

Diese sechs Arten von Plug-Ins müssen bestimmten Implementierungsregeln folgen. Die von verschiedenen Plug-In-Typen implementierten Funktionen unterscheiden sich.

Das Objekt „DialogServer“

Als Toolbox für Plug-In-Implementierer stellt Dialogue Server eine Schnittstelle für die Unterstützung verschiedener Funktionen bereit. Diese Schnittstelle ist automatisch während der Laufzeit für Skript-Plug-Ins durch das Objekt **DialogServer** verfügbar.

Plug-In-Installation

Plug-Ins werden in Dialogue Admin installiert. Skript-Plug-Ins können in Dialogue Admin auch bearbeitet werden. Weitere Informationen finden Sie unter **Plug-In-Repository**.

Hinweis: In der vorliegenden Dokumentation liegt der Fokus auf dem Skriptansatz zum Schreiben von Plug-Ins anstatt auf der Implementierung der Plug-Ins als COM-Objekte. Code-Beispiele verwenden die Sprache *JScript*.

Kundendaten-Plug-Ins

Kundendaten-Plug-Ins

Kundendaten enthalten mit Kunden verbundene Informationen und Daten aus unterschiedlichen Quellen. In einigen Domänen sind alle Daten schreibgeschützt. In anderen können Kundendaten durch Dialogue Server hinzugefügt oder bearbeitet werden. Dialogue Server unterstützt zwei Funktionen, um Kundendaten zu aktualisieren.

- **Automatische Aktualisierungen**

  Dialogue Server generiert automatisch SQL-Anweisungen, um Änderungen in der Datenbank weiterzugeben.

- **Von Plug-Ins verarbeitete Aktualisierungen**

  Ein Plug-In verarbeitet die Aktualisierung.

Andererseits wird ein *Kundendaten-Plug-In* implementiert, um Änderungen einer Datengruppe in einer Kundendomäne zu speichern.

Kundendaten-Plug-Ins implementieren die Schnittstelle **IMHDataGroupPlugin**.

Funktionen in einem Kundendaten-Plug-In

Das Kundendaten-Plug-In muss drei Arten von Aktualisierungen verarbeiten können: *Einfügen*, *Aktualisieren* und *Löschen*. Daher müssen drei Funktionen implementiert werden. Zusätzlich kann die optionale Methode *CheckForDublicates(...)* implementiert werden, um eine Prüfung auf Duplikate vornehmen zu können.

Im folgenden Beispiel werden die Personeninformationen in einer Datenbank aktualisiert (das Kundendatenbankschema BALDER):

```
function InsertRow( Customer, Fields )
{
```

```
   var PersonID = DialogServer.SQLRetrieveValueRaw(
   "SELECT balder.idnr_seq.nextval FROM dual", null, null);
   PersonIDArray = new Array(1);
   PersonIDArray[0] = PersonID;

   DialogServer.SQLExecuteRaw(
  "INSERT INTO balder.idnr (idnr, id_type) values (:idnr, 'P')", PersonIDAr-
ray, null);

   DialogServer.SQLExecuteRaw(
      "INSERT INTO balder.person (person_id, lastname, firstname) " +
      "values (:person_id, :lastname, :firstname)",
      new Array(PersonID,
      Fields.FieldBySourceName("lastname").NewValue,
      Fields.FieldBySourceName("firstname").NewValue),
      null);

   return PersonID;
}


function UpdateRow( Customer, Fields )
{
   var SqlStatement = "UPDATE balder.person SET lastname = :lastname, " +
   "firstname = :firstname " +
   "where person_id = :person_id";

   DialogServer.SQLExecuteRaw(
      SqlStatement,
      new Array(Fields.FieldBySourceName("lastname").NewValue,
      Fields.FieldBySourceName("firstname").NewValue,
      Fields.FieldBySourceName("person_id").OldValue),
      null);
}


function DeleteRow( Customer, Fields )
{
   PersonIDArray = new Array(1);
   PersonIDArray[0] = Fields.FieldBySourceName("person_id").OldValue;

   DialogServer.SQLExecuteRaw(
   "DELETE FROM balder.pers_cat_relation where person_id = :person_id",
      PersonIDArray,
      null);

   DialogServer.SQLExecuteRaw(
   "DELETE FROM balder.person where person_id = :person_id",
      PersonIDArray,
      null);

   DialogServer.SQLExecuteRaw(
   "DELETE FROM balder.idnr where idnr = :person_id",
      PersonIDArray,
      null);
}

function CheckForDuplicates(Customer, DuplicateList )
{
//Do some duplicate checking by searching in the database for potential
//duplicates
```

```
//Check firstname and lastname
SQLParams = new Array(2);
SQLParams[0] = Customer.FieldValue("Firstname");
SQLParams[1] = Customer.FieldValue("LastName");
DuplicateRecordSet = DialogServer.SQLOpen("Contacts_DupCheck_1", SQLParams);
while (!DuplicateRecordSet.EOF)
{
DuplicateList.AddDuplicate( DuplicateRecordSet.Fields("cust_char_id").Value
 );

      DuplicateRecordSet.MoveNext();

}

   //Check email address
SQLParams = new Array(1);
SQLParams[0] = Customer.FieldValue("Email");
if ((SQLParams[0] != null) && (SQLParams[0] != ""))
{
DuplicateRecordSet = DialogServer.SQLOpen("Contacts_DupCheck_2", SQLParams);
while (!DuplicateRecordSet.EOF)
{
 DuplicateList.AddDuplicate( DuplicateRecordSet.Fields("cust_char_id").Value
 );
 DuplicateRecordSet.MoveNext();
}
}

   //Check addresses - Loop through all address types of customer
Customer.FirstDetailData("Addresses");
while (!Customer.DetailDataEof("Addresses"))
{
SQLParams = new Array(2);
SQLParams[0] = Customer.FieldValue("Addresses.Streetname");
SQLParams[1] = Customer.FieldValue("Addresses.Streetno");
DuplicateRecordSet = DialogServer.SQLOpen("Contacts_DupCheck_3", SQLParams);
while (!DuplicateRecordSet.EOF)
{
 DuplicateList.AddDuplicate( DuplicateRecordSet.Fields("pa_pers_cust_id").Value
 );
   DuplicateRecordSet.MoveNext();
}
Customer.NextDetailData("Addresses");
}
}
```

Hinweis: Das Skript verwendet das Objekt **DialogServer**, um SQL-Anweisungen auszuführen. Dieses Objekt ist als „Toolbox“ immer verfügbar, wenn Plug-Ins implementiert werden.

Verzweigungs-Plug-Ins

Verzweigungs-Plug-Ins

Verzweigungs-Plug-Ins implementieren die Logik hinter Operationen in Dialogen. Sie werden von Dialogue Server aufgerufen, wenn Operationen ausgeführt werden. Der Name *Verzweigungs-Plug-In* wird verwendet, da eine Operation mehrere Verzweigungen haben kann. Die Plug-Ins entsprechen den Funktionen hinter *einer Verzweigung*.

Es gibt viele installierte Standardoperationstypen, und die meisten dieser Operationen werden mittels Skript implementiert. Daher ist der Code für jeden frei zugänglich, um ihn als Beispiel für das Schreiben benutzerdefinierter Verzweigungs-Plug-Ins zu verwenden.

Schnittstellendefinition

Verzweigungs-Plug-Ins implementieren die Schnittstelle **IMHBranchPlugin**.

Funktionen in einem Verzweigungs-Plug-In

Ein einfaches Verzweigungs-Plug-In kann so aussehen (unter Verwendung von JScript):

```
function ExecuteBranch(BranchInfo, Participants)
{
   // Get parameters of branch
   FilterSQL = BranchInfo.ParamValue("FilterSQL");
   IncludeContext = BranchInfo.ParamValue("IncludeContext");

   Participants.SelectBySQL(FilterSQL.Name, null, IncludeContext);
   BranchInfo.ReportProgressStatus('Filtering participants using SQL', 0);
   Participants.AcceptAll();
   BranchInfo.ReportProgressStatus('Completed filtering', 100);
}

function GetParamDefs(ParamDefs)
{
   ParamDefs.AddParam("FilterSQL", "Filter SQL", 9, true, null, "", "");
   ParamDefs.AddParam("IncludeContext", "Use context from SQL", 6, true,
false, "", "");
}
```

Das Plug-In muss zwei Funktionen implementieren: *ExecuteBranch* und *GetParamDefs*.

Die zweite Funktion *GetParamDefs* beschreibt die Parameter, die für die Ausführung des Plug-Ins benötigt werden. In dieser Funktion werden Parameter zusammen mit gewissen Eigenschaften wie Datentyp, Standardwerte usw. hinzugefügt.

Die erste Funktion *ExecuteBranch* enthält den Code, der innerhalb der Operation ausgeführt wird. In diesem Beispiel verwendet das Plug-In SQL, um Teilnehmer für die Von-Gruppe auszuwählen.

Bestimmte Verzweigungs-Plug-Ins implementieren noch eine dritte Funktion: *GetDynamicList*. Sie wird verwendet, um eine Liste von Parameterwerten zu füllen, die der Benutzer bei der Einrichtung der Verzweigung auswählen kann:

```
function GetDynamicList(CustDomainID, ParamName, DynamicList)
{
   if (ParamName == "MyListParam")
   {
     // Add the fields to in the matrix to polulate
     DynamicList.AddField("ID", "integer");
     DynamicList.AddField("Name", "string");

     //Populate matrix
     DynamicList.AddRow( new Array(1, "Options1") );
     DynamicList.AddRow( new Array(2, "Options2") );
     DynamicList.AddRow( new Array(3, "Options3") );
   }
}
```

Ausdrucks-Plug-Ins

Ausdrucks-Plug-Ins

Ausdrucks-Plug-Ins implementieren benutzerdefinierte Ausdrucksfunktionen. Jedes in Dialogue Admin hinzugefügte Ausdrucks-Plug-In stellt eine Ausdrucksfunktion dar.

Schnittstellendefinition

Ausdrucks-Plug-Ins implementieren die Schnittstelle **IMHExprFunctionPlugin**.

Implementieren eines Ausdrucks-Plug- Ins

Ein Beispiel eines Ausdruck-Plug-Ins könnte wie folgt aussehen (Verwendung von JScript):

```
// An example of an user defined expression function
//
// string RepeatString(InputValue, RepeatTimes)
//
// Repeats a given string n times

function DefineFunction( ExprFunctionDef )
{
   ExprFunctionDef.Description = "Repeats a string N times";

   ExprFunctionDef.ResultType = "string";

   ExprFunctionDef.AddParam("InputValue", "string");
   ExprFunctionDef.AddParam("RepeatTimes", "integer");
}


function EvaluateFunction( Customer, InputValue, RepeatTimes )
```

```
{
   var RepeatedString = "";

   for (var i=1; i<=RepeatTimes; i++)
   {
       RepeatedString = RepeatedString + InputValue;
   }


   return RepeatedString;
}
```

Generische Plug-Ins

Generische Plug-Ins

Generische Plug-Ins implementieren benutzerdefinierte Logik, die über Dialogue Server ausgeführt wird. Diese Plug-Ins werden durch Dialogue Server aufgerufen, typischerweise als Antwort auf die generische API-Methode *ExecutePlugin*.

Schnittstellendefinition

Generische Plug-Ins implementieren die Schnittstelle **IMHGenericPlugin**.

Implementieren eines generischen Plug-Ins

Ein generisches Plug-In kann beispielsweise so aussehen (verwendet JScript):

```
// The Execute method might be declared with any list of input parameters
function Execute(CustomerName, CustomerID)
{
   // Get parameters of branch
   var CallingUser = DialogServer.User.UserName;

   return "Plug-in called by: " + CallingUser + ", CustomerName= " + Custo-
merName;
}
```

Nachrichten-Plug-Ins

Nachrichten-Plug-Ins

Nachrichten-Plug-Ins implementieren die Erstellung von Nachrichten. Nachrichten ist ein allgemeiner Begriff für unterschiedliche Dokumenttypen, die bei der Kommunikation mit Kunden verwendet werden. Nachrichten-Plug-Ins werden aufgerufen, um Dokumente zu erstellen, bevor sie dem Kunden gesendet werden.

Es gibt drei installierte Standardnachrichten-Plug-Ins:

- **Text zusammenfügen**

  Fügt eine Textvorlage mit Kundendaten zusammen. Das resultierende Dokument kann eine E-Mail (Text oder HTML), SMS oder Dateien zum Exportieren sein.

- **MS Word zusammenfügen**

  Fügt eine MS Word-Vorlage, die Seriendruckfelder enthält, mit Kundendaten zusammen.

- **XSL-Transformation**

  Benötigt eine Vorlage, die eine XSLT-Formatvorlage enthält. Dieses Plug-In ruft Kundendaten von Dialogue Server als XML ab und führt eine XSL-Transformation aus, um das Ausgabedokument zu erstellen.

Schnittstellendefinition

Nachrichten-Plug-Ins implementieren die Schnittstelle **IMHCreateMessagePlugin**. Nachrichten-Plug-Ins können ebenfalls die neueren Schnittstellen **IMHCreateMessagePlugin2** und **IMHCreateMessage-Plugin3** implementieren. Das ist jedoch nicht erforderlich.

Funktionen in einem Nachrichten-Plug-In

Es gibt vier Funktionen, die in einem Nachrichten-Plug-In implementiert werden müssen.

ProduceMessages erstellt Nachrichten für Kunden im Objekt *CustomerContainer.* Manchmal wird eine einzelne Nachricht erstellt und für jeden einzelnen Kunden getrennt gespeichert (z. B. E-Mails). In anderen Fällen wird nur eine große Nachricht von Dialogue Server abgefragt (eine Exportdatei). Daher müssen Nachrichten-Plug-Ins beide Methoden unterstützen.

ProduceSingleMessage ähnelt „ProduceMessages“. Aber anstelle des Objekts *CustomerContainer*, das mehrere Kunden enthält, wird das Objekt *Customer* als Eingabeparameter verwendet, das nur einen Kunden enthält. Somit erstellt diese Funktion immer nur eine einzelne Ausgabenachricht.

Die Funktion *SetDataFields* teilt Dialogue Server einen Satz von Kundendatenfeldern mit, die benötigt werden, um eine Nachricht für eine gegebene Vorlage zu erstellen. *CanSetDataFields* teilt mit, ob das Plug-In Kundendatenfelder in Folgendem bestimmen kann: *SetDataFields.*

Ausgangskanal-Plug-Ins

Ausgangskanal-Plug-Ins

Ausgangskanal-Plug-Ins werden verwendet, um zu implementieren, wie die elektronische Kommunikation (E-Mail, SMS usw.) stattfindet. Dialogue Server ruft Ausgangskanal-Plug-Ins auf, um Nachrichten durch Kommunikationsprotokolle senden zu können.

Es gibt ein auf dem System installiertes Standard-Plug-In für das Versenden von E-Mails ebenso wie ein Beispiel für das Implementieren des Versendens von SMS-Nachrichten.

Schnittstellendefinition

Ausgangskanal-Plug-Ins implementieren die Schnittstelle **IMHOutputChannelPlugin**.

Funktionen eines Ausgangskanal-Plug-Ins

Es gibt zwei Funktionen, die in einem Ausgangskanal-Plug-In implementiert werden: *SendMessages* und *OutputChannel*:

```
function SendMessages( OutputChannel, OutputMessages )
{
   //Let's start...
   OutputChannel.ReportProgressStatus("Starting to send SMS messages", 0);

   //Get communication param values
   ServerIP = OutputChannel.ControlParamValue("ServerIP");
   ServerPort = OutputChannel.ControlParamValue("ServerPort");
   UserName = OutputChannel.ControlParamValue("UserName");
   Pwd = OutputChannel.ControlParamValue("Password");
   FromNumber = OutputChannel.ControlParamValue("FromNumber");

   //Create the server object
   SmsServer = new ActiveXObject("SMSServiceProvider.SmsCtrl");

   //Connect to the SMS server
   if (! SmsServer.Connect(ServerIP + ":" + ServerPort, UserName, Pwd,
   "MHS_CLIENT_FROM_CPAGW", "MHS_CLIENT_TO_CPAGW"))
   {
      throw new Error(0, "Could not connect to SMS Server");
   }

   //For each message...
   while (! OutputMessages.MemberEof)
   {
      //Get control parameters distinct to each message
      Message = OutputMessages.Message;
      MessageText = Message.ContentText;

      Cellular = Message.ControlParamValue("Cellular");

      //Try to send the message
      if (SmsServer.SendMsg(FromNumber, Cellular, MessageText, "text", 0,
0))
      {
         //Set status OK for sent messages (and remove them from queue)
         OutputMessages.SetStatusOK();
      } else
      {
         //Mark messages that could not be sent with an error message
         ErrorText = "SMS send error: " + SmsServer.GetErrorString();
         OutputMessages.SetStatusError(ErrorText);
         DialogServer.LogDebugMessage(ErrorText);
      }

      //Go to next message
      OutputMessages.NextMember();
   }

   //Completed!
```

```
   OutputChannel.ReportProgressStatus("Completed sending messages", 100);
}

function GetControlParamDefs( ControlParamDefs )
{
   //Commuication params
   ControlParamDefs.AddParam("ServerIP", "100.100.100.0", true, false);
   ControlParamDefs.AddParam("ServerPort", "15000", true, false);
   ControlParamDefs.AddParam("UserName", "user", true, false);
   ControlParamDefs.AddParam("Password", "pass", true, false);
   ControlParamDefs.AddParam("FromNumber", "2008", true, false);

   //Params to be merge per message
   ControlParamDefs.AddParam("Cellular", "", true, true);
}
```

SendMessages enthält den Code, der die eigentliche Nachricht versendet, die als Eingabe für das Objekt *OutputMessages* weitergeleitet wird. *OutputChannel* ist ein Objekt, das den Zugriff auf *Steuerparameter* bereitstellt.

Steuerungsparameter

Steuerparameter sind Parameter, die vom Plug-In benötigt werden, um Nachrichten zu versenden. Das Plug-In definiert seine eigenen Steuerparameter in *GetControlParamDefs*.

Steuerparameterwerte werden vom Benutzer eingerichtet, wenn Vorlagen und Basisnachrichten erstellt werden. Steuerparameter können Seriendruckfelder enthalten, die es ermöglichen, Parameterwerte gleich den Kundendatenfeldern festzulegen.

Der Parameter für die Mobiltelefonnummer („Cellular“) im obigen Skript kann beispielsweise als das Seriendruckfeld „Person.Cellular“ festgelegt werden, wenn eine SMS-Vorlage erstellt wird.

Typen

Einige Typen sind definiert, z. B. Daten- und Parametertypen.

Verzweigungsparametertypen

Einleitung

Verzweigungsparametertypen sind die verschiedenen, in Dialogue konfigurierbaren Parametertypen, wenn eine Verzweigung innerhalb einer Operation eingerichtet wird. Einige dieser Datentypen sind einfach, wie z. B. Ganzzahlen oder Zeichenfolgen, andere dagegen repräsentieren Systemobjekte, wie z. B. eine Dialoggruppe oder einen Fragebogen.

Parametertypen

In der folgenden Tabelle werden die verschiedenen Parametertypen definiert.

ID	Logischer Typ	Zurückgegebener Typ	Gespeicherter Typ	GDS-aktiviert	ExtraInfo1	ExtraInfo2
0	Text	Zeichenfolge	Zeichenfolge	Ja		
1	UNC (Dateipfad)	Zeichenfolge	Zeichenfolge	Nein	Standarderweiterung (optional). Siehe Erklärung unten.	
2	Datum (nur Datum)	Datum/Uhrzeit	Datum/Uhrzeit	Ja	Min. Wert (optional)	Max. Wert (optional)
3	Integer	ganze Zahl	ganze Zahl	Ja	Min. Wert (optional)	Max. Wert (optional)
4	Gleitkommazahl	Fließkomma	Fließkomma	Ja	Min. Wert (optional)	Max. Wert (optional)
5	Prozentzahl	Fließkomma	Fließkomma	Ja	Min. Wert (optional)	Max. Wert (optional)
6	Kontrollkästchen	boolean	Zeichenfolge	Ja		
7	Basisnachricht	**IMHUnmergedMessage**	ganze Zahl	Ja	Auf *einen* Kanal beschränken, z. B. „EMAIL“ (optional)	
8	Fragebogen	**IMHQuestionnaire**	ganze Zahl	Ja	Auf **true** (wahr) festlegen, um alle Fragebögen aufzulisten; auf **false** (falsch) festlegen, um nur identifizierte oder Fragebögen im gemischten Modus aufzulisten. Siehe Erklärung unten.	
9	SQL-Definition	**IMHSQLDef**	ganze Zahl	Nein		
10	Dialoggruppe	**IMHDialogGroup**	ganze Zahl	Nein		

ID	Logischer Typ	Zurückgegebener Typ	Gespeicherter Typ	GDS-aktiviert	ExtraInfo1	ExtraInfo2
11	Optionsfelder	Zeichenfolge	Zeichenfolge	Nein	Durch Semikolon getrennte Liste anzuzeigender Namen	Durch Semikolon getrennte Liste zu speichernder Namen
12	Telemarketing-Projekt	**IMHCCProject**	ganze Zahl	Nein		
13	Aufrufstatusliste	**IMHCCCallStatusList**	Zeichenfolge	Nein		
14	Dynamische Liste	Zeichenfolge	Zeichenfolge	Nein	Name des zu speichernden und des anzuzeigenden Feldes, durch Semikolon getrennt. *Siehe Beispiel weiter unten.*	Beschriftung des Auswahldialogfeldes, z. B. „Einen Wert auswählen“.
15	Auswahl	**IMHSelection**	ganze Zahl	Ja		
16	Aufgabenarbeitsgruppe	ganze Zahl	ganze Zahl	Nein		
17	Ordner	Zeichenfolge	Zeichenfolge	Nein		
18	Aktivitätstyp	Zeichenfolge	Zeichenfolge	Nein		
19	Kanaltyp	Zeichenfolge	Zeichenfolge	Nein		
20	Kategorie	**IMHCategory**	Zeichenfolge	Nein	Kategorietyp. Mögliche Werte sind „SIMPLE“, „BLOCKING“, „SCORING“ und „VALUES“.	
21	E-Mail-Unzustellbar-	**IMHEmailBounceCodeList**	Zeichenfolge	Nein		

ID	Logischer Typ	Zurückgegebener Typ	Gespeicherter Typ	GDS-aktiviert	ExtraInfo1	ExtraInfo2
	keitscodes					
22	Dynamische Liste mit Mehrfachauswahl	**IMHDynamicValuesList**	Zeichenfolge	Nein	Name des zu speichernden und des anzuzeigenden Feldes, durch Semikolon getrennt.	Beschriftung des Auswahldialogfeldes, z. B. „Mindestens einen Wert auswählen“.
23	Berichtsvorlage	**MHReportTemplate**	ganze Zahl	Nein		
24	Berichtsformat	**IMHReportFormat**	ganze Zahl	Nein		
25	Kundensortierfelder	**IMHCustomerSortFieldList**	Zeichenfolge	Nein		
26	Auswahlkriterium	Zeichenfolge	Zeichenfolge	Nein	Boolescher Wert, der angibt, ob der Benutzer den Kriteriumsausdruck als Text bearbeiten kann oder nicht.	Der interne Klassenname des Editors, der zur Erstellung eines neuen Kriteriums verwendet werden muss. Falls leer, dient der Domänenfeld-Kriteriumseditor als Standard.
27	PSS-Entscheidungsoptimierung	ganze Zahl	ganze Zahl	Nein		
28	PSS-Entscheidungs-	Zeichenfolge	Zeichenfolge	Nein		

ID	Logischer Typ	Zurückgegebener Typ	Gespeicherter Typ	GDS-aktiviert	ExtraInfo1	ExtraInfo2
	ergebnis					
29	PSS-Angebote (mehrfach)	**IMHCmsIdList**	Zeichenfolge	Ja		
30	PSS-Angebot (einfach)	ganze Zahl	ganze Zahl	Ja		
31	PSS-Behandlung	ganze Zahl	ganze Zahl	Ja		
32	PSS-Verhaltenstyp	ganze Zahl	ganze Zahl	Ja		
33	PSS-Antwortindikator	ganze Zahl	ganze Zahl	Ja		
34	Kundenliste	**IMHCustomerList**	ganze Zahl	Ja		

ID ist eine Zahl, die zur Bezeichnung des Parametertyps verwendet wird, wenn Parameter in den *GetParamDefs*-Methoden des Verzweigungs-Plug-Ins definiert werden.

Logischer Typ ist eine Beschreibung des Parametertyps.

Zurückgegebener Typ ist der zurückgegebene Datentyp oder die zurückgegebene Schnittstelle, wenn Parameterwerte in der *ExecuteBranch*-Methode des Verzweigungs-Plug-Ins abgefragt werden.

Gespeicherter Typ ist der intern von Dialogue Server verwendete Datentyp, um Parameterwerte zu speichern.

GDS-aktiviert: Wenn der Verzweigungsparametertyp in der unterstützten Dialogeinrichtung verfügbar ist.

ExtraInfo1 und *ExtraInfo2* sind Zeichenfolgen, die bei der Definition von Parametern in den *GetParamDefs*-Methoden des Verzweigungs-Plug-Ins festgelegt werden. Dadurch werden Dialogue Server und

Dialogue zusätzliche Informationen über die Behandlung des Parameters zur Verfügung gestellt. Die Bedeutung dieser Zeichenfolgen ist je nach Parametertyp unterschiedlich.

Festlegen der Standarderweiterungen

Der UNC-Parametertyp ermöglicht es, Standarderweiterungen festzulegen. Die im Plug-In festgelegten Standarderweiterungen werden von Dialogue verwendet, wenn der Benutzer bei der Einrichtung der Operation aufgefordert wird, den Dateinamen anzugeben.

Das Format der Standarderweiterung wird definiert als:

<Beschreibung1>|<Erweiterung1>|<Beschreibung2>|<Erweiterung2>|...

Mehrere Erweiterungen können durch ein Semikolon getrennt werden. Siehe Beispiel 2 unten.

Beispiel 1: Textdateien|*.txt

Beispiel 2: Alle Bilddateien|*.jpg;*.gif|JPG-Dateien|*.jpg|GIF-Dateien|*.gif|Bitmap-Bilddateien|*.bmp

Beachten Sie, dass das Element „Alle Dateien|*.*“ automatisch von Visual Dialogue hinzugefügt wird.

Festlegen der aufzulistenden Fragebögen

Der Fragebogen-Parametertyp ermöglicht Ihnen die Festlegung der Fragebögen, die dem Benutzer von Visual Dialogue zur Auswahl gestellt werden.

Wenn „TRUE“ (WAHR) festgelegt wurde, werden alle Fragebögen in der Datenbank aufgelistet.

Wenn „FALSE“ (FALSCH) oder kein Wert festgelegt wurde, werden ausschließlich Fragebögen der Umfragetypen *Gemischt* oder *Identifiziert* aufgelistet. Fragebögen, die auf eine andere Kundendomäne als die vom Dialog verwendete Kundendomäne beschränkt sind, werden ebenfalls herausgefiltert.

Reservierte Parameternamen

Vom Verzweigungs-Plug-In werden die Namen der entsprechenden Parameter definiert. Allerdings sind einige Parameternamen vom System reserviert und haben eine spezielle Bedeutung:

Parametername	Beschreibung
mh_copy_to_remote_group	Wird von der Verzweigung “In anderen Dialog kopieren” verwendet, um einen entsprechenden Teilnehmerablauf als Pfeile in Dialog Designer zu visualisieren.
mh_copy_from_remote_group	Wird von den Verzweigungen “Mithilfe einer Remotegruppe auswählen” und “In anderen Dialog kopieren” verwendet, um einen entsprechenden Teilnehmerablauf als Pfeile in Dialog Designer zu visualisieren.
mh_telemarketing	Wird von der Verzweigung “Durch Telemarketing verarbeitet” verwendet, um die gezielt im Telemarketing-Projekt anzusprechenden Teilnehmer (in der Absendergruppe) zu identifizieren.

Hinweis: Obwohl lediglich drei Parameternamen in der aktuellen Version reserviert sind, werden zukünftig möglicherweise mehr Parameternamen reserviert werden. Die Benennung von Parametern mit “mh_” am Anfang wird nicht empfohlen.

Der „dynamic list“-Parametertyp

Der *dynamic list*-Parametertyp ermöglicht es dem Programmierer, einen Parameter mit einer Reihe von Werten hinzuzufügen, aus denen der Visual Dialogue-Benutzer auswählen kann. Im folgenden Beispiel wird gezeigt, wie dies gemacht wird.

Beispiele

Das folgende Beispiel stellt die Verwendung von Parametern in einem Verzweigungs-Plug-In dar.

```
function ExecuteBranch(BranchInfo,
 Participants)
{
   //Get parameter values
   CategoryName = BranchInfo.ParamValue("BCat");
   CategoryOption = BranchInfo.ParamValue("Option");
   ..............
}

function GetParamDefs( ParamDefs )
{
   //Define parameters needed in this branch. One string parameter and one
dynamic list parameter
   ParamDefs.AddParam("Option", "Option",11,true,null, "Set Category; Remove
 Category" , "S;R");
   ParamDefs.AddParam("BCat", "Simple Category", 14, true, null, "cat_na-
me;cat_name", "Choose Simple Category");
}

function GetDynamicList( CustDomainID, ParamName, DynamicList)
{
  //Populate the list from which the user can select values in Visual Dialogue

  DynamicList.PopulateFromSQL("SelectCategories", Params);
}
```

Fragebogen-Datentypen

Einleitung

Eine Reihe von durch Fragen und Alternativen in Fragebögen verwendeten Datentypen.

Datentypen

Die Tabelle unten beschreibt die unterschiedlichen Datentypen.

Datentypname	Beschreibung
Zeichenfolge	Zeichenfolge-Datentyp
ganze Zahl	Ganzzahl-Datentyp
Fließkomma	Gleitkomma-Datentyp
Datum/Uhrzeit	Datum/Zeit-Datentyp

date	Datum-Datentyp (enthält nicht die Zeitinformation aus „datetime“)
boolean	boolescher Datentyp

Schnittstellen

COM-Schnittstellen sind in der Plug-In-API definiert, um Plug-Ins mit in Dialogue Server enthaltenen Funktionen auszustatten.

Diese Schnittstellen werden sowohl in Skript-Plug-Ins als auch in COM-Implementierungen verwendet.

IMHAnswerForm-Schnittstelle

Beschreibung

IMHAnswerForm bietet Zugriff auf ein Antwortformular.

Anwendung

IMHAnswerForm wird verwendet, um die Antworten eines Antwortformulars zu lesen und festzulegen.

Übergeordnete Schnittstelle

„IMHAnswerForm“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

datetime **AnsweredTimestamp**

Das Datum und die Uhrzeit, zu dem das Antwortformular vom Kunden beantwortet wurde. Schreibgeschützt.

int64 **AnswerFormID**

Die eindeutige ID des Antwortformulars in der Datenbank (QRY_ANSWER_FORM.QAF_ID).

string **ArchiveRef**

Ein Zeichenfolgenwert, der optional verwendet werden kann. Schreibgeschützt.

variant **BroadcastID**

Die ID der Übertragung, die mit dem Antwortformular verbunden ist. „BroadcastID“ ist *Null*, wenn keine Übertragung damit verbunden ist. Schreibgeschützt.

string **ChannelName**

Der Name des Kanals, über den das Antwortformular empfangen wurde.

bool **Completed**

Zeigt an, ob das Antwortformular abgeschlossen wurde. Schreibgeschützt.

string **Context**

Der beim Beantworten des Fragebogens verwendete Kontext. Schreibgeschützt.

integer CustDomainID

Die ID der Kundendomäne, zu welcher der Kunde gehört, der den Fragebogen beantwortet.

string **CustomerID**

Die eindeutige ID des Kunden, der den Fragebogen beantwortet. Schreibgeschützt.

bool **IsAnonymous**

Zeigt an, ob das Antwortformular eine anonyme oder identifizierte Antwort ist. Schreibgeschützt.

IMHQuestionnaire Questionnaire

Der mit dem Antwortformular verbundene Fragebogen. Schreibgeschützt.

int64 **ParticipantID**

Die ID des Dialogteilnehmers, der den Fragebogen beantwortet. Schreibgeschützt.

Hinweis: Diese Information ist optional und nur vorhanden, wenn ein Antwortformular mit einem Dialogteilnehmer verbunden ist.

datetime **ProcessedTimestamp**

Zeigt an, ob und wann das Antwortformular als verarbeitet markiert wurde. Schreibgeschützt.

Hinweis: „ProcessedTimestamp“ ist für die benutzerdefinierte Verwendung konzipiert und kann festgelegt oder gelöscht werden, indem die Methode *MarkAsProcessed( )* aufgerufen wird.

datetime **ScanTimestamp**

Das Datum und die Uhrzeit, zu der das Antwortformular gescannt wurde. Ist *null*, falls nicht verwendet. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **ClearAllAnswers**( )

Löscht alle Antworten des Antwortformulars.

void **ClearAllComments**( )

Löscht alle Abschnittskommentare des Antwortformulars.

void **ClearAnswer**(*string* QuestionKey, *string* AlternativeKey)

Löscht beliebige Antworten einer angegebenen Frage oder Alternative.

Falls keine Alternative angegeben werden soll, setzen Sie *AlternativeKey* auf eine leere Zeichenfolge. In diesem Fall werden die Antworten der ganzen Frage gelöscht, einschließlich der Antworten aller Alternativen in der Frage.

void **ClearComment**(*integer Section*No)

Löscht den Kommentar des angegebenen Abschnitts.

void **ClearState**( )

Löscht den Status des Antwortformulars.

void **Delete**( )

Löscht das Antwortformular aus der Datenbank.

variant **GetAnswer**(*string* QuestionKey, *string* AlternativeKey)

Gibt den geantworteten Wert der angegebenen Frage oder Alternative zurück. Falls keine Alternative angegeben werden soll, setzen Sie *AlternativeKey* auf eine leere Zeichenfolge.

Der Datentyp des zurückgegebenen Werts hängt vom Datentyp der Frage oder Alternative im Fragebogen ab.

string **GetAnswerSingleChoice**(*string* QuestionKey)

Gibt die Antwort auf eine Einzelauswahlfrage zurück. Der zurückgegebene Wert entspricht *AlternativeKey* für die ausgewählte Alternative.

string **GetAnswerSingleChoiceText**(*string* QuestionKey)

Gibt die Antwort auf eine Einzelauswahlfrage zurück. Der zurückgegebene Wert ist die Beschriftung der ausgewählten Alternative.

string **GetComment**(*integer Section*No)

Gibt den Kommentar des angegebenen Abschnitts zurück.

bool **IsAnswered**(*string* QuestionKey, *string* AlternativeKey)

Gibt *true* zurück, wenn die angegebene Frage oder Alternative beantwortet wurde (oder bei Mehrfachauswahlfragen ausgewählt wurde). Anderenfalls wird *false* zurückgegeben.

void **MakeAnonymous**( )

Anonymisiert ein Antwortformular, indem der Bezug zu einem Kunden entfernt wird.

void **MarkAsProcessed**(*bool* Unmark)

Markiert ein Antwortformular als verarbeitet, indem die Eigenschaft *ProcessedTimestamp* festgelegt wird. Diese Eigenschaft entspricht der Datenbankspalte QRY_ANSWER_FORM.QAF_PROCESSED_TIMESTAMP.

Das Setzen von *Unmark* auf *true* löscht den Zeitstempel, anstatt ihn festzulegen.

Hinweis: Unter „**IMHQuestionnaireInterface**-Schnittstelle“ im Abschnitt zur Methode *GetAnonymous-AnswerForm( )* wird erklärt, wie diese Methode verwendet werden kann.

void **MarkAsProcessedInDialog**(**IMHBranchInfo** Branch, *bool* Unmark)

Markiert ein Antwortformular als im Dialog verarbeitet. Welche Antwortformulare im Dialog verarbeitet wurden, wird in der Datenbankspalte DLG_PROCESSED_ANSWER_FORM gespeichert.

Branch ist ein Objekt, das die Verzweigung im Dialog darstellt, der das Antwortformular verarbeitet hat.

Das Setzen von *Unmark* auf *true* löscht den Zeitstempel, anstatt ihn festzulegen.

Hinweis: Unter „**IMHDialog**-Schnittstelle“ im Abschnitt zur Methode *GetAnonymousAnswerForm( )* wird erklärt, wie diese Methode verwendet werden kann.

void **RelateToCustomer**(**IMHCustomer** Customer)

Verbindet eine anonyme Antwort mit einem identifizierten Kunden.

Customer ist ein Objekt, das den Kunden darstellt.

void **RelateToCustomerByID**(*integer* CustDomainID, *string* CustomerID, *string* Context)

Bewirkt dasselbe wie *RelateToCustomer( )*. Allerdings werden als Parameter IDs anstelle eines Objekts verwendet, das den Kunden darstellt.

CustDomainID ist die ID der Kundendomäne, zu welcher der Kunde gehört.

CustomerID ist die eindeutige ID des Kunden, der den Fragebogen beantwortet hat.

Context ist der Kontext, in dem der Kunde den Fragebogen beantwortet hat. „Context“ wird in einem Dialog verwendet und dient der Unterscheidung zwischen mehreren Teilnehmern mit derselben Kunden-ID.

void **SetAnswer**(*string* QuestionKey, *string* AlternativeKey, *variant* AnswerValue)

Legt den Wert der angegebenen Frage oder Alternative fest.

Der Datentyp von *AnswerValue* muss dem Datentyp der angegebenen Frage oder Alternative entsprechen.

void **SetAnswerBool**(*string* QuestionKey, *string* AlternativeKey, *bool* AnswerValue)

Legt den Wert einer angegebenen Alternative in einer Mehrfachauswahlfrage fest.

void **SetComment**(*integer* Section*No*, *string* Comment)

Legt den Kommentar des angegebenen Abschnitts fest.

Kommentare

Die Begriffe *QuestionKey, AlternativeKey* und *SectionNo* werden im Folgenden erläutert:

QuestionKey ist eine Zeichenfolge, welche die Frage identifiziert, und ist eindeutig innerhalb eines Fragebogens. *QuestionKey* wird einer Frage automatisch zugewiesen, wenn sie im Fragebogen-Designer

in Visual Dialogue in einen Fragebogen eingefügt wird. Sie hat folgendes Standardformat: Q1, Q2, Q3, ..., Q *n*. Der Benutzer kann *QuestionKey* jedoch bearbeiten und z. B. den Schlüssel „VNAME“ für die Frage „Vorname“ verwenden.

QuestionKey wird auf Fragen angewendet. Für Alternativen in einer Frage verwendet man *AlternativeKey*. *AlternativeKey* ist innerhalb einer Frage eindeutig. Das Standardformat für „AlternativeKey“ ist: A1, A2, A3, ..., A*n*.

SectionNo ist die Nummer des Abschnitts, beginnend mit „1“ für den ersten Abschnitt des Fragebogens.

Beispiele

Das folgende Beispiel zeigt, wie man Teilnehmer und ihre Antwortformulare in einer Operation durchgeht. Die Teilnehmer werden abhängig von ihren Antworten in die Empfängergruppe verschoben.

```
 function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   //Open the container
   Participants.Open("mh_customer_id");

   while (! Participants.MemberEof) {

     //Get the answer forms of questionnaire with id 1048
     var AnswerForm = Participants.Participant.GetAnswerForm(1048);

if (AnswerForm != Null)
{
//Dependent of their answers move participants to the to-group
     if ((AnswerForm.IsAnswered("Q10", "A3") || (AnswerForm.IsAnswered("Q10",
 "A4"))
     Participants.Participant.Accept();
}

     //Move to next participants
     Participants.NextMember();
   }
   ..............
   }
```

IMHAnswerFormList-Schnittstelle

Beschreibung

IMHAnswerFormList bietet Zugriff auf eine Liste oder einen Container von Antwortformularen.

Anwendung

„IMHAnswerFormList“ wird verwendet, um einen Satz von Antwortformularen zu iterieren.

Übergeordnete Schnittstelle

„IMHAnswerFormList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Gibt die Anzahl der Antwortformulare in der Liste zurück.

bool **MemberEof**

Boolesche Eigenschaft, die anzeigt, ob es keine weiteren Antwortformulare in der Liste gibt. Schreibgeschützt.

IMHAnswerForm AnswerForm

Bietet Zugriff auf das aktuelle Antwortformular.

Methoden

Die folgenden Methoden werden unterstützt:

bool **NextMember** ( )

Macht das nächste Antwortformular in der Liste zum aktuellen. Gibt *true* zurück, wenn es keine weiteren Antwortformulare im Container gibt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHBranchDynamicList-Schnittstelle

Beschreibung

IMHBranchDynamicList bietet Zugriff auf ein Objekt, das eine Liste der möglichen Werte für einen Verzweigungsparameter enthält.

Anwendung

„IMHBranchDynamicList“ wird in einem *Verzweigungs-Plug-In* verwendet, wenn eine Liste der Parameterwerte erstellt wird, aus denen der Benutzer in Visual Dialogue auswählen kann. Diese Schnittstelle wird mit dem Parametertyp *Dynamic List* verwendet.

Verwenden Sie entweder den Aufruf von *AddField/AddRow* oder eine der Füllmethoden, um eine Liste der Werte zu erstellen.

Übergeordnete Schnittstelle

„IMHBranchDynamicList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddField**(*string* FieldName, *string* Datatype)

Fügt ein Feld oder eine Spalte zu einer Liste von Werten hinzu. Rufen Sie *AddField* mehrfach auf, um mehrere Spalten einzufügen. Rufen Sie dann *AddRow* auf, um die eigentlichen Werte hinzuzufügen.

FieldName ist der Name des Feldes oder der Spalte.

Datatype ist eine Zeichenfolge, welche die gültigen **Datentypen** angibt.

void **AddRow**(*array* Values)

Fügt eine Zeile zu einer Liste von Werten hinzu.

Values ist ein Array mit der Anzahl der Elemente, die den Feldern entsprechen, die über *AddField* definiert wurden.

void **PopulateFromSQL**(*string* SQLName, *array* Params)

Füllt die Liste durch eine SQL-SELECT-Anweisung.

SQLName ist der Name einer SQL-Anweisung, die in Dialogue Admin im SQL-Repository gespeichert ist.

Params ist ein Array, das die in die SQL-Anweisung einzubindenden Parameterwerte enthält.

void **PopulateFromXML**(*string* XMLData)

Füllt die Liste durch ein XML-Dokument.

XMLData ist eine Zeichenfolge, die die verwendete XML enthält, um die Liste zu füllen. Die XML folgt der Standard-XML-Darstellung eines Datensatzes in Microsoft .NET.

Beispiele

Das folgende Beispiel illustriert die Verwendung eines dynamischen Listenparameters.

```
function ExecuteBranch(BranchInfo, Participants)
{
   //Get parameter values
   CategoryName = BranchInfo.ParamValue("BCat");

   ..............
}

function GetParamDefs( ParamDefs )
{
   //Define parameters needed in this branch. One string parameter and one
dynamic list parameter
   ParamDefs.AddParam("BCat", "Simple Category", 14, true, null, "cat_na-
me;cat_name", "Choose Simple Category");
```

```
}

function GetDynamicList(CustDomainID, ParamName, DynamicList)
{
  //Populate the list from which the user can select values in Visual Dialogue

  if (ParamName = "BCat")
  {
    DynamicList.PopulateFromSQL("SelectCategories", Params);
  }
else if (ParamName == "xml_param")
{
   var fs = new ActiveXObject("Scripting.FileSystemObject");
   var XMLFile = fs.OpenTextFile("PopulateFromXML.xml", 1, false, -1); //Un-
icode XML file
   var XMLDataString = XMLFile.ReadAll();

   DynamicList.PopulateFromXML(XMLDataString);
}
}
```

IMHBranchInfo-Schnittstelle

Beschreibung

IMHBranchInfo ermöglicht den Zugriff auf eine Information über eine Verzweigung einer Operation in einem Dialog.

Anwendung

„IMHBranchInfo“ wird in Verzweigungs-Plug-Ins während der Ausführung von Operationen verwendet.

Übergeordnete Schnittstelle

„IMHBranchInfo“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **CustDomainID**

Gibt die ID (eindeutige Kennung) der aktuellen Kundendomäne zurück. Schreibgeschützt.

IMHDialog Dialog

Ermöglicht den Zugriff auf ein Objekt, das für den Dialog steht, in der die Verzweigung existiert. Schreibgeschützt.

IMHDialogGroup FromGroup

Ermöglicht den Zugriff auf ein Objekt, das für die Absendergruppe der aktuellen Operation steht. Schreibgeschützt.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Verzweigung. Schreibgeschützt.

string **Name**

Name ist der Name der Verzweigung.

IMHDialogOperation Operation

Ermöglicht den Zugriff auf ein Objekt, das für die Operation steht. Schreibgeschützt.

variant **ParamValue**[*string* ParamName]

Gibt einen Verzweigungsparameterwert zurück, der in Visual Dialogue festgelegt wurde. Der zurückgegebene Datentyp hängt vom Typ des Parameters ab. Eine Reihe von **Parametertypen** ist zur Verwendung in Verzweigungs-Plug-Ins definiert.

ParamName ist der Name des Parameters entsprechend der Definition des Verzweigungs-Plug-Ins in der Funktion *GetParamDefs*.

integer **Priority**

Ermöglicht den Zugriff auf die Priorität der Verzweigung. Eine Operation enthält möglicherweise mehrere Verzweigungen, und *priority* steht für die Ausführungsreihenfolge der Verzweigungen in der Operation. Die erste Verzweigung hat den Prioritätswert „1“. Schreibgeschützt.

IMHDialogGroup ToGroup

Ermöglicht den Zugriff auf ein Objekt, das für die Empfängergruppe der Verzweigung steht. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **CheckForAbort**( )

Überprüft, ob ein Abbruch der aktuellen Ausführung signalisiert wurde. In einem solchen Fall wird vom Dialogue Server eine Ausnahme ausgelöst.

void **ReportProgressStatus**(*string* StatusText, *integer* PercentCompleted)

Berichtet dem Dialogue Server bei Ausführung einer Verzweigung den entsprechenden Status und Fortschritt.

StatusText ist eine Zeichenfolge, die den gegenwärtigen Status beschreibt.

PercentCompleted ist die Prozentangabe, die den Fortschritt des Sendevorgangs angibt. Gültig sind Werte zwischen 0 und 100.

Hinweis: *ReportProgressStatus( )* überprüft, ob ein Abbruch der Operation signalisiert wurde. *CheckForAbort( )* wird implizit aufgerufen.

Hinweis: *StatusText* unterstützt die Übersetzung (Lokalisierung). Der Übersetzungsmechanismus ist ein interner Mechanismus, der für die Lokalisierung der Standard-Skript-Plug-Ins verwen-

det wird und nicht für benutzerdefinierte oder Add-On-Plug-Ins anwendbar ist. Siehe auch die *Translate(..)*- und *TranslateAndFormat(..)*-Methoden von **IMHDialogServer**.

void **ReportProgressStatusFormat**(*string* StatusText, *array* Params, *integer* PercentCompleted)

Berichtet dem Dialogue Server bei Ausführung einer Verzweigung den entsprechenden Status und Fortschritt. Wie *ReportProgressStatus(...)* weiter oben, jedoch wird hier die Formatierung von *StatusText* unterstützt.

StatusText ist eine Zeichenfolge, die den gegenwärtigen Status beschreibt.

Params ist ein Array von Werten, das während des Formatierens in die Zeichenfolge (*Msg*) eingefügt wird.

PercentCompleted ist die Prozentangabe, die den Fortschritt des Sendevorgangs angibt. Gültig sind Werte zwischen 0 und 100.

Formatierungsbeispiel (JScript):

ReportProgressStatusFormat("Aktueller Status: Element {0} von {1} wird bearbeitet", new array(5, 10), 50);

Die Textplatzhalter {0} und {1} werden mit Werten aus dem Array *Params* (Formatierung) ersetzt. Dies führt zu folgendem Ergebnis:

„Aktueller Status: Element 5 von 10 wird bearbeitet“

Beispiele

Es sind keine Beispiele verfügbar.

IMHBranchParamDefs-Schnittstelle

Beschreibung

IMHBranchParamDefs bietet Zugriff auf ein Objekt, das die Definition der von der Verzweigung benötigten Parameter enthält.

Anwendung

Auf „IMHBranchParamDefs“ wird über Verzweigungs-Plug-Ins zugegriffen, um Parameter zu definieren.

Übergeordnete Schnittstelle

„IMHBranchParamDefs“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddParam**(*string* Name, *string* DisplayLabel, *integer* ParamType, *bool* Required, *variant* DefaultValue, *variant* ExtraInfo1, *variant* ExtraInfo2)

Definiert und fügt eine Parameterdefinition zum Verzweigungs-Plug-In hinzu. Dieser Parameter wird vom Benutzer in Visual Dialogue konfiguriert, wenn eine Operation eingerichtet wird.

Name ist der technische Name des Parameters, der ihn identifiziert.

DisplayLabel ist der dem Benutzer angezeigte Parametername.

ParamType ist der Parametertyp. Es ist ein Satz von **Parametertypen** vordefiniert.

Required legt fest, ob der Parameter einen Wert haben muss, bevor die Verzweigung ausgeführt werden kann.

DefaultValue ist ein optionaler Parameter, der den Standardwert angibt. Falls keiner verwendet wird, setzen Sie „DefaultValue“ auf einen *null*-Bezug.

ExtraInfo1 und *ExtraInfo2* sind Werte, die bei manchen Parametertypen verwendet werden.

Hinweis: *DisplayLabel* unterstützt die Übersetzung (Lokalisierung). Der Übersetzungsmechanismus ist ein interner Mechanismus, der für die Lokalisierung der Standard-Skript-Plug-Ins verwendet wird und nicht für benutzerdefinierte oder Add-On-Plug-Ins anwendbar ist. Siehe auch die *Translate(..)*- und *TranslateAndFormat(..)*-Methoden von **IMHDialogServer**.

Beispiele

Im Beispiel unten wird gezeigt, wie auf das Objekt *ParamDefs* durch die IMHBranchParamDefs-Schnittstelle zugegriffen wird.

```
function GetParamDefs( ParamDefs )
{
   //Define parameters needed in this branch. One string parameter and one
dynamic list parameter
   ParamDefs.AddParam("Option", "Option",11,true,null, "Set Category; Remove
 Category" , "S;R");
   ParamDefs.AddParam("BCat", "Simple Category", 14, true, null, "cat_na-
me;cat_name", "Choose Simple Category");

}
```

IMHBranchPlugin-Schnittstelle

Beschreibung

IMHBranchPlugin enthält Methoden, die von Verzweigungs-Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Das Verhalten von Verzweigungen in einer Dialogoperation wird durch „IMHBranchPlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHBranchPlugin“ erbt von der Basisschnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **ExecuteBranch**(**IMHBranchInfo** BranchInfo, **IMHParticipants** Participants)

„ExecuteBranch“ wird von Dialogue Server aufgerufen, wenn eine Operation ausgeführt wird, um Teilnehmer in der Von-Gruppe der Operation zu handhaben.

BranchInfo enthält Informationen zur ausgeführten Verzweigung, z. B. wie die Verzweigung in Visual Dialogue konfiguriert ist.

Participants ist ein Objekt, das die Dialogteilnehmer in der Von-Gruppe der Operation darstellt.

void **GetDynamicList**(*integer* CustDomainID, *string* ParamName, **IMHBranchDynamicList** DynamicList)

„GetDynamicList“ wird von Dialogue Server aufgerufen, um mögliche Werte eines Parameters des Typs *Dynamic List* zu erhalten.

CustDomainID ist die ID der aktuellen Kundendomäne.

ParamName ist der Name des Parameters, wie in *GetParamDefs* definiert.

DynamicList ist ein Objekt, das die Liste der zu füllenden Werte darstellt.

void **GetParamDefs**(**IMHBranchParamDefs** ParamDefs)

„GetParamDefs“ wird von Dialogue Server aufgerufen, um die Definition der vom Verzweigungs-Plug-In benötigten Parameter zu erhalten.

ParamDefs ist ein Objekt, das einen Satz von Parametern darstellt, die definiert werden.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCategory-Schnittstelle

Beschreibung

IMHCategory ermöglicht den Zugriff auf Informationen über eine Auswahl.

Anwendung

IMHCategory wird in Plug-Ins zum Zugriff auf Auswahlen verwendet.

Übergeordnete Schnittstelle

IMHCategory erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Name**

Ermöglicht den Zugriff auf den eindeutigen Namen der Kategorie. Schreibgeschützt.

string **Value**

Ermöglicht den Zugriff auf den eindeutigen Wert der Kategorie, wenn ein Wert für die Kategorie festgelegt wurde. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCacheItem-Schnittstelle

Beschreibung

IMHCacheItem ist eine Basisschnittstelle zum Zwischenspeichern, die in der Plug-In-API benutzt wird.

Anwendung

IMHCacheItem wird von Plug-Ins implementiert, die Objekte zwischenspeichern. Beachten Sie, dass diese Schnittstelle momentan nur in Verbindung mit **IMHMessageAssembleInfo** verwendet wird.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ItemName**

Der eindeutige Name des zwischengespeicherten Elements. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCCCallStatusList-Schnittstelle

Beschreibung

IMHCCCallStatusList ermöglicht den Zugriff auf eine Liste der Aufrufstatustypen.

Anwendung

„IMHCCCallStatusList“ wird in Plug-Ins zum Zugriff auf Aufrufstatustypen von Telemarketing-Projekten verwendet. Außerdem wird sie vom standardmäßigen „Select/Divide“ von Aufrufstatus-Verzweigungs-Plug-Ins genutzt.

Übergeordnete Schnittstelle

„IMHCCCallStatusList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ListAsString** Ermöglicht den Zugriff auf eine durch Komma getrennte Liste von Aufrufstatus-IDs. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCCProject-Schnittstelle

Beschreibung

IMHCCProject ermöglicht den Zugriff auf Informationen über ein Telemarketing-Projekt.

Anwendung

„IMHCCProject“ wird in Plug-Ins zum Zugriff auf Telemarketing-Projekte verwendet.

Übergeordnete Schnittstelle

„IMHCCProject“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID des Telemarketing-Projekts. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHChannelMessage-Schnittstelle

Beschreibung

„IMHChannelMessage“ bietet aus einem Ausgangskanal-Plug-In Zugriff auf eine Nachricht.

Anwendung

„IMHChannelMessage“ wird in Plug-Ins als eine Schnittstelle zu einer *Nachricht* verwendet, die über einen Ausgangskanal versendet wird. Eine Kanalnachricht repräsentiert eine einzelne Nachricht.

Übergeordnete Schnittstelle

„IMHChannelMessage“ erbt von der Basisschnittstelle **IMHMessage**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **IsContentBinary**

Gibt an, ob der Inhalt der Nachricht binär codiert oder Text ist.

string **FileEncoding**

FileEncoding ist der Name der Dateicodierung, die für die aktuelle Nachricht verwendet wird. Diese Einstellung wird nur auf Textdateien angewendet.

string **FileExtension**

Diese Zeichenfolge wird als Dateierweiterung verwendet, wenn Nachrichten des aktuellen Typs als Datei gespeichert werden (z. B. *htm* oder *txt*).

integer **MessageBundleID**

Die eindeutige Kennung des Nachrichtenpakets in der Datenbank (entspricht der Datenbankspalte MESSAGE_BUNDLE_LOG.MBL_ID).

string **MessageName**

Der Name, welcher der Nachricht gegeben wird.

int64 **MessageOutputQueueID**

Die eindeutige Kennung des Elements in der Nachrichtenwarteschlange der Datenbank (entspricht der Datenbankspalte MESSAGE_OUTPUT_QUEUE.MOQ_ID). Schreibgeschützt. Intern vom Nachrichten-Framework verwendet.

bool **OnePerCustomer**

Gibt an, ob das Nachrichtenpaket (zu dem die Nachricht gehört) eine Nachricht pro Kunde (z. B. eine E-Mail) oder eine Nachricht für alle Kunden (z. B. eine exportierte Liste aller Kunden) enthält.

bool **StoredAsFile**

Gibt an, ob der Nachrichteninhalt in einer Datei gespeichert wird oder nicht.

Hinweis: Wenn die Nachricht in einer Datei gespeichert wird, verweist die Eigenschaft *MessageUNC* auf diese. Wenn die Nachricht nicht in einer Datei gespeichert wird, enthält eine der beiden Eigenschaften *ContentText* (Inhalt ist Text) oder *ContentBinary* (Inhalt ist binär codiert) den Nachrichteninhalt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageContainer-Schnittstelle

Beschreibung

„IMHMessageContainer“ bietet Zugriff auf einen Satz von Nachrichten.

Anwendung

„IMHMessageContainer“ wird im Ausgangskanal-Plug-In als Schnittstelle zu einem Satz von Nachrichten verwendet. Wenn Nachrichten versandt werden, empfängt das Plug-In ein Containerobjekt, das die zu versendenden Nachrichten enthält.

Übergeordnete Schnittstelle

„IMHMessageContainer“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **ContainsPersistentMessages**

Gibt an, ob der Container Nachrichten enthält, die (beständig) in Dialogue Database gespeichert oder dynamisch generiert und nur temporär gespeichert sind. Schreibgeschützt.

integer **Count**

Gibt die Anzahl der Nachrichten im Container zurück.

bool **MemberEof**

Dieser boolesche Wert gibt an, ob keine weiteren Nachrichten im Container vorhanden sind. Schreibgeschützt.

IMHChannelMessage Message

Bietet Zugriff auf die aktuelle Nachricht.

integer **ProcessedCount**

Die Anzahl der bereits vom Plug-In verarbeiteten Nachrichten. Dies bezeichnet alle Nachrichten, die mit *SetStatusOK* oder *SetStatusError* markiert wurden. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

bool **NextMember** ( )

Setzt die nächste Nachricht im Container als aktuelle Nachricht. Gibt *True* zurück, wenn keine weitere Nachricht im Container vorhanden ist.

void **SetStatusError**(*string* ErrorMessages)

Markiert die aktuelle Nachricht mit einem Fehler. Diese Methode wird vom Plug-In aufgerufen, wenn die Nachricht nicht gesendet werden konnte.

ErrorMessage ist eine Zeichenfolge, welche die Fehlerursache angibt.

void **SetStatusOK**( )

Wird vom Plug-In aufgerufen, um Dialogue Server den erfolgreichen Versand der aktuellen Nachricht mitzuteilen. Dies entfernt die Nachricht aus der Sendewarteschlange.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCmsIdList-Schnittstelle

Beschreibung

IMHCmsIdList ermöglicht den Zugriff auf eine Liste mit Ganzzahl-ID-Werten.

Anwendung

„IMHCmsIdList“ wird in Plug-Ins zum Zugriff auf eine Liste mit ID-Werten in der PSR-Datenbank verwendet.

Übergeordnete Schnittstelle

„IMHCmsIdList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Id**[*integer* Index]

Ermöglicht den Zugriff auf einen einzelnen ID-Wert.

Index ist der Index des ID-Werts in der Liste. Ein Indexwert von 0 steht für den ersten Wert.

integer **Count**

Gibt die Anzahl der ID-Werte in der Sammlung zurück.

Methoden

Die folgenden Methoden werden unterstützt:

void **Add**(*integer* Id)

Fügt einen ID-Wert an das Ende der Liste hinzu.

void **Clear**( )

Löscht die Liste der ID-Werte.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCreateMessagePlugin-Schnittstelle

Beschreibung

IMHCreateMessagePlugin enthält Methoden, die von Nachrichten-Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die Logik, die eine Nachricht basierend auf einer Vorlage oder Basisnachricht erstellt wird durch „IMH-CreateMessagePlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHCreateMessagePlugin“ erbt von der Basisschnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

bool **CanSetDataFields**( )

Der zurückgegebene Wert von „CanSetDataFields“ teilt Dialogue Server mit, ob das Plug-In die Kundendatenfelder bestimmen kann, die zum Erstellen einer Nachricht benötigt werden. Falls der Wert *true* ist, wird die Methode *SetDataFields* von Dialogue Server aufgerufen.

IMHMessageBundle **ProduceMessages**(**IMHUnmergedMessage** UnmergedMessage, **IMHCustomerContainer** CustomerContainer)

„ProduceMessages“ wird von Dialogue Server aufgerufen, um einen Satz von Nachrichten zu erstellen. „ProduceMessages“ gibt ein Objekt zurück, das ein Nachrichtenpaket darstellt. Ein solches Objekt kann durch Verwendung von Methoden des Objekts *CustomerContainer* erstellt werden.

UnmergedMessage ist ein Objekt, das die Nachrichtenvorlage oder die Basisnachricht darstellt.

CustomerContainer ist ein Objekt, das die Kunden darstellt, für welche die Nachrichten erstellt werden. Das Plug-In wird typischerweise alle Kunden in *CustomerContainer* durchschleifen.

IMHMessage **ProduceSingleMessage**(**IMHUnmergedMessage** UnmergedMessage, **IMHCustomer** Customer)

„ProduceSingleMessage“ wird von Dialogue Server aufgerufen, um *eine einzelne* mit einem Kunden verbundene Nachricht zu erstellen. „ProduceSingleMessage“ gibt ein Objekt zurück, das eine Nachricht darstellt. Ein solches Objekt kann durch Verwenden von Methoden des Objekts *Customer* erstellt werden.

UnmergedMessage ist ein Objekt, das die Nachrichtenvorlage oder die Basisnachricht darstellt.

Customer ist ein Objekt, das den Kunden darstellt, für den die Nachricht erstellt wird.

void **SetDataFields**(**IMHUnmergedMessage** UnmergedMessage, *string* FieldsAvailable)

„SetDataFields“ wird von Dialogue Server aufgerufen, wenn der verwendete Satz von Kundendatenfeldern (aus der Kundendomäne) bekannt sein muss, um die Nachricht zu erstellen. Daher wird durch die Implementierung dieser Methode das Plug-In mitteilen, wenn zum Erstellen von Nachrichten ein Satz von Datenfeldern benötigt wird. Diese Nachrichten verwenden die Vorlage, die definiert ist durch *UnmergedMessage.* Das Plug-In aktualisiert die Eigenschaft *DataFields* von *UnmergedMessage.*

UnmergedMessage ist ein Objekt, das die Nachrichtenvorlage oder die Basisnachricht darstellt.

FieldsAvailable ist eine durch Semikola getrennte Liste der in der Kundendomäne verfügbaren Felder. *FieldsAvailable* kann beispielsweise dazu verwendet werden, zu verifizieren, dass nur erlaubte Felder in der Nachrichtenvorlage verwendet werden.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCreateMessagePlugin2-Schnittstelle

Beschreibung

IMHCreateMessagePlugin2 erweitert **IMHCreateMessagePlugin**, das Methoden enthält, die von Nachrichten-Plug-Ins implementiert werden. Alle Nachrichten-Plug-Ins müssen **IMHCreateMessagePlugin** implementieren. Die Implementierung von „IMHCreateMessagePlugin2“ und **IMHCreateMessagePlugin3** ist dagegen optional.

Durch die Implementierung von „IMHCreateMessagePlugin2“ hat Dialogue Server die Möglichkeit, große Nachrichtenerstellungen in mehrere Pakete aufzuteilen und so die Nachrichtenerstellung zu optimieren.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die Logik, die eine Nachricht basierend auf einer Vorlage oder eine Basisnachricht erstellt, wird durch „IMHCreateMessagePlugin2“ implementiert.

Übergeordnete Schnittstelle

„IMHCreateMessagePlugin2“ erbt von der Basisschnittstelle der Nachrichten-Plug-Ins: **IMHCreateMessagePlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHMessageBundle ProduceMessagesEx(**IMHUnmergedMessage** UnmergedMessage, **IMHCustomerContainer** CustomerContainer, *integer* MaxBundleSize)

„ProduceMessagesEx“ wird von Dialogue Server aufgerufen, um einen Satz von Nachrichten zu erstellen. „ProduceMessages“ gibt ein Objekt zurück, das ein Nachrichtenpaket darstellt. Ein solches Objekt kann durch Verwendung von Methoden des Objekts *CustomerContainer* erstellt werden.

UnmergedMessage ist ein Objekt, das die Nachrichtenvorlage oder die Basisnachricht darstellt.

CustomerContainer ist ein Objekt, das die Kunden darstellt, für welche die Nachrichten erstellt werden. Das Plug-In wird typischerweise alle Kunden in *CustomerContainer* durchschleifen.

MaxBundleSize ist die maximale Anzahl der zu produzierenden Nachrichten. Das Plug-In produziert nur diese Anzahl von Nachrichten, auch wenn *CustomerContainer* eine größere Anzahl von Kunden enthalten kann.

Hinweis: Nach der Erstellung jeder Nachricht muss das Plug-In *CustomerContainer.NextMember()* aufrufen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCreateMessagePlugin3-Schnittstelle

Beschreibung

IMHCreateMessagePlugin3 erweitert **IMHCreateMessagePlugin**, das Methoden enthält, die von Nachrichten-Plug-Ins implementiert werden. Alle Nachrichten-Plug-Ins müssen **IMHCreateMessagePlugin** implementieren. Die Implementierung von „IMHCreateMessagePlugin2“ und **IMHCreateMessagePlugin3** ist dagegen optional.

Durch die Implementierung von „IMHCreateMessagePlugin3“ kann das Nachrichten-Plug-In die Art optimieren, wie Nachrichten in der Datenbank gespeichert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die Logik, welche die Speicherung von Nachrichten optimiert wird durch „IMHCreateMessagePlugin3“ implementiert.

Übergeordnete Schnittstelle

„IMHCreateMessagePlugin3“ erbt von der Basisschnittstelle der Nachrichten-Plug-Ins: **IMHCreateMessagePlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **AssembleMessage**(**IMHMessage** UnassembledMessage, **IMHMessageAssembleInfo** AssembleInfo)

Diese Methode wird aufgerufen, wenn der Server das Plug-In auffordert, eine Nachricht zusammenzufügen. Dies wird typischerweise durchgeführt, bevor eine Nachricht durch einen Kanal versendet wird (z. B. das Versenden einer E-Mail) oder eine Nachricht einem Benutzer angezeigt wird.

Das Plug-In ändert typischerweise die Eigenschaften *ContentText* oder *ContentBinary* von *UnassembledMessage*, um den Inhalt der endgültigen Nachricht zu enthalten.

void **PrepareAssembleInfo**(**IMHMessageAssembleInfo** AssembleInfo)

Diese Methode wird von *AssembleMessage(...)* aufgerufen, damit das Plug-In das Zusammenfügen der Nachrichten für ein spezifisches Paket vorbereiten kann. Dies bedeutet, dass das Plug-In das Objekt *AssembleInfo* für den späteren Gebrauch bei vielfachem Aufruf der Methode *AssembleMessage(...)* vorbereiten kann. *AssembleInfo* wird vom Server zwischengespeichert, um den vielfachen Aufruf der Methode *AssembleMessage(...)* zu optimieren.

void **PrepareUnassembledContent**(**IMHMessageBundle** MessageBundle)

Diese Methode wird aufgerufen, wenn ein neues Nachrichtenpaket erstellt wird. Sie wird aufgerufen, bevor eine Nachricht erstellt wird, und lässt das Nachrichten-Plug-In entscheiden, ob die für das Paket erstellten Nachrichten nicht zusammengefügt sind. Das Plug-In kann auch einen beliebigen Inhalt vorbereiten, der gemeinsam für alle Nachrichten im Paket gespeichert wird.

Beispiele

Es sind keine Beispiele verfügbar.

IMHContentItem-Schnittstelle

Beschreibung

IMHContentItem bietet Zugriff auf ein Inhaltselement, das in einem Inhaltsobjekt verwendet wird.

Anwendung

IMHContentItem wird bei der Implementierung von Regelskripten in Inhaltsobjekten verwendet. Siehe auch **IMHContentObjectRulePlugin**.

Übergeordnete Schnittstelle

IMHContentItem erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **AnchorEnabled**

Gibt an, ob Inhaltselemente vom Typ *image* eine zugeordnete URL besitzen.

bool **AnchorNewWindow**

Zeigt an, ob die in *AnchorUrl* angegebene URL in einem neuen Browserfenster geöffnet werden soll.

bool **AnchorTracking**

Zeigt an, ob die Verknüpfungsverfolgung verwendet werden soll, um zu protokollieren, wann der Benutzer ein Bild anklickt und die in *AnchorUrl* angegebene Webseite geöffnet wird.

string **AnchorUrl**

Wenn *AnchorEnabled* auf *True* gesetzt ist, enthält *AnchorUrl* die Zieladresse, die geöffnet wird, wenn der Benutzer auf das Bild klickt.

bool **ContentTagsEnabled**

Gibt an, ob Zusammenführungs-Tags unterstützt werden.

string **ContentText**

Enthält den Ursprungsinhalt für Inhaltselemente der Typen *Text* und *HTML*.

string **ContentType**

Der Inhaltstyp. Mögliche Werte sind:

Wert	Beschreibung
Bild	Der Inhalt ist ein Bild, dass in *ContentUrl* angegeben ist.
html	Der Inhalt ist ein Stück HTML-Code, der in *ContentText* gespeichert ist.
Text	Der Inhalt ist reiner Text, der in *ContentText* gespeichert ist.
message_template	Der Inhalt wird mit einer Nachrichtenvorlage erstellt. Die Vorlage wird in *MessageTemplateID* angegeben.
url	Der Inhalt ist eine HTML-Seite, die in *ContentUrl* angegeben ist.

string **ContentUrl**

Enthält die URL, die den eigentlichen Inhalt der Inhaltselemente vom Typ *image* und *url* enthält.

bool **ContentUrlIsWebPublicFile**

Gibt an, ob „ContentUrl“ auf eine veröffentlichte Datei verweist. Dieser Wert ist schreibgeschützt.

string **Desc**

Ein Text, der das Inhaltselement beschreibt.

string **Key**

Eine Zeichenfolge, die das Inhaltselement identifiziert. Dieser Wert ist innerhalb eines Inhaltsobjekts eindeutig.

integer **MessageTemplateID**

Die ID der Nachrichtenvorlage, wenn der Inhalt den Typ *message_template* hat.

string **Name**

Der Name des Inhaltselements.

double **RandomWeighting**

Ein Gewichtungsfaktor, der bei der zufälligen Auswahl eines Inhaltselements berücksichtigt wird, wenn eine Regel darauf konfiguriert ist.

Methoden

Es werden keine Methoden unterstützt

Beispiele

Es sind keine Beispiele verfügbar.

IMHContentItemList-Schnittstelle

Beschreibung

IMHContentItemList bietet Zugriff auf eine Liste der Inhaltselemente eines Inhaltsobjekts.

Anwendung

„IMHContentItemList“ wird bei der Implementierung von Skriptregeln in Inhaltsobjekten verwendet. Siehe auch **IMHContentObjectRulePlugin**.

Übergeordnete Schnittstelle

IMHContentItemList erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Die Gesamtanzahl der Inhaltselemente in der Liste.

IMHContentItem **ContentItem**[*variant* KeyOrIndex]

Bietet Zugriff auf ein Inhaltselement in der Liste.

KeyOrIndex ist entweder die Kennung eines Inhaltselements (Zeichenfolge) oder dessen Index (Ganzzahl) in der Liste. Die Indizierung beginnt mit 1 für das erste Element.

Methoden

Die folgenden Methoden werden unterstützt:

IMHContentItem **AddContentItem**( )

Erstellt ein neues Inhaltselement und hängt es am Ende der Liste an.

Diese Methode ermöglicht es, ein Inhaltselement dynamisch in einer Skriptregel zu erstellen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHContentObjectRulePlugin-Schnittstelle

Beschreibung

IMHContentObjectRulePlugin enthält Methoden, die von Skriptregeln in Inhaltsobjekten implementiert werden.

Anwendung

Durch „IMHCreateMessagePlugin“ wird die Logik für eine Skriptregel in einem Inhaltsobjekt implementiert.

Übergeordnete Schnittstelle

IMHOutputChannelPlugin erbt von der Basisschnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

variant **EvaluateRule**(**IMHContentItemList** ContentItems, **IMHCustomer** Customer, **IMHContentRequestParams** RequestParams)

EvaluateRule(...) wird aufgerufen, wenn eine Skriptregel in einem Inhaltsobjekt evaluiert werden muss.

Der Benutzer sollte bei der Implementierung dieser Funktion einen booleschen Wert (*True* oder *False*) oder ein Inhaltselement zurückgeben. Die Rückgabe eines booleschen Wertes bedeutet, dass die Regel erfolgreich ausgewertet wird und dass das angegebene Inhaltselement, das für diese Regel anzuwenden ist, das Ergebnis der Ausführung des Inhaltsobjekts ist. Die Rückgabe eines Inhaltselements bedeutet auch, dass diese Regel erfolgreich ausgewertet wird, dass aber das zurückgegebene Inhaltselement das Ergebnis der Ausführung des Inhaltsobjektes ist.

ContentItems ist eine Liste aller Inhaltselemente, die im aktuellen Inhaltsobjekt definiert sind.

Customer ist ein Objekt, das den Kunden darstellt, für den das Inhaltsobjekt ausgeführt wird. „Customer“ ist *null*, wenn das Inhaltsobjekt im anonymen Modus ausgeführt wird.

IMHContentObjectUtilsRequestParams ist eine Liste zusätzlicher Parameter, die zum Zeitpunkt der Ausführung des Inhaltsobjekts angegeben werden.

Beispiele

Es sind keine Beispiele verfügbar.

IMHContentObjectUtils-Schnittstelle

Beschreibung

IMHContentObjectUtils bietet Zugriff auf Inhaltsobjekte und verwandte Aufgaben und Objekte.

Anwendung

„IMHContentObjectUtils“ wird verwendet, um intern Inhaltsobjektlogik in die Nachrichtenproduktion zu implementieren.

Übergeordnete Schnittstelle

„IMHContentObjectUtils“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

string **GetContentObjectLogicForMessageMerge(** *string* ContentObjectKey, *string* RequestParams, *bool* RenderAsHtml **)**

Erweitert die Logik eines Inhaltsobjekts und gibt diese als IF-Anweisungen und Funktionsaufrufe für die Verwendung innerhalb einer Nachrichtenvorlage zurück.

ContentObjectKey ist der eindeutige Schlüssel des Inhaltsobjekts.

RequestParams ist eine Zeichenfolge, die zusätzliche Parameter beim Aufrufen der Inhaltsobjekte enthält.

RenderAsHtml gibt an, ob das Ergebnis dieses Methodenaufrufs in einer HTML-Nachricht (z. B. einer HTML-E-Mail) verwendet werden soll oder nicht (z. B. eine Text-E-Mail).

string **GetDataFieldsInContentObject(** *integer* CustDomainID, string ContentObjectKey **)**

Gibt eine durch Semikola getrennte Liste der für die Ausführung eines Inhaltsobjekts benötigten Domänenfelder zurück.

CustDomainID ist die eindeutige ID der Kundendomäne.

ContentObjectKey ist der eindeutige Schlüssel des Inhaltsobjekts.

Beispiele

Es sind keine Beispiele verfügbar.

IMHContentRequestParams-Schnittstelle

Beschreibung

IMHContentRequestParams bietet Zugriff auf zusätzliche Parameter, wenn ein Inhaltsobjekt ausgeführt wird. Die Parameter sind als Name-Wert-Parametersammlung strukturiert.

Anwendung

„IMHContentRequestParams“ wird bei der Implementierung von Skriptregeln in Inhaltsobjekten verwendet. Siehe auch **IMHContentObjectRulePlugin**.

Übergeordnete Schnittstelle

„IMHContentRequestParams“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Die Gesamtanzahl der Parameter in der Liste.

string **Name**[*integer* Index]

Ermöglicht unter Angabe des Indexes Zugriff auf den Namen eines einzelnen Parameters. Der erste Parameter hat den Index 0.

string **Value**[*string* Name]

Ermöglicht unter Angabe des Parameternamens Zugriff auf den Wert eines einzelnen Parameters.

Methoden

Die folgenden Methoden werden unterstützt:

integer **IndexOfName**(*string* Name)

Durchsucht die Liste nach dem Namen eines einzelnen Parameters und gibt dessen Index in der Liste zurück. Der erste Parameter hat den Index 0.

Beispiele

Es sind keine Beispiele verfügbar.

IMHControlParamDefs-Schnittstelle

Beschreibung

IMHControlParamDefs bietet Zugriff auf ein Objekt, das die Definition der benötigten Parameter zum Versenden von Nachrichten enthält.

Anwendung

Auf „IMHControlParamDefs“ wird von Ausgangskanal-Plug-Ins zugegriffen, um Steuerparameter zu definieren.

Übergeordnete Schnittstelle

„IMHControlParamDefs“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddParam**(*string* Name, *string* DefaultValue, *bool* Required, *bool* IsMergeParam)

Definiert und fügt Steuerparameter für ein Ausgangskanal-Plug-In hinzu.

Name ist der technische Name des Parameters, der ihn identifiziert.

DefaultValue ist ein optionaler Parameter, der den Standardwert angibt.

„Required“ gibt an, ob dieser Parameter beim Versenden von Nachrichten erforderlich ist.

IsMergeParam gibt an (falls *false* gesetzt ist), dass der Parameter auf der Kanalebene eingerichtet ist, d. h. bei der Konfiguration eines *Ausgangskanals*. Falls *true* gesetzt ist, wird der Parameter eingerichtet, wenn Vorlagen und Basisnachrichten erstellt werden, und kann Seriendruckfelder enthalten, die sich auf Kundendomänendaten beziehen.

void **AddParamEx**(*string* Name, *string* DefaultValue, *bool* Required, *bool* IsMergeParam, *bool* TestSendOverride)

Bewirkt dasselbe wie „AddParam()“. Allerdings wird ein weiterer Parameter benötigt: *TestSendOverride.*

TestSendOverride gibt an, ob der Benutzer den Wert der Steuerparameter ändern kann, wenn eine Nachricht in Visual Dialogue zu Testzwecken versendet wird.

Hinweis: In der aktuellen Version kann nur ein Steuerparameter überschrieben werden, wenn eine Nachricht zum Testen versendet wird. Wenn mehr als ein Parameter *TestSendOverride* auf *true* gesetzt hat, wird nur der erste Parameter in Visual Dialogue im Fenster *Testversandoptionen* angezeigt. Die Werte anderer Parameter werden gelöscht, wenn diese Option festgelegt ist.

Beispiele

Das untere Beispiel zeigt, wie Steuerparameter in einem Plug-In definiert werden, das den SMS-Kanal implementiert.

```
function GetControlParamDefs( ControlParamDefs )
{
   //Communication params
   ControlParamDefs.AddParam("ServerIP", "100.150.20.254", true, false);
   ControlParamDefs.AddParam("ServerPort", "15000", true, false);
   ControlParamDefs.AddParam("FromNumber", "2008", true, false);

   //Params to be merge per message
   ControlParamDefs.AddParam("Cellular", "", true, true);
}
```

IMHCustomer-Schnittstelle

Beschreibung

IMHCustomer ermöglicht den Zugriff auf einen einzelnen Kunden.

Anwendung

IMHCustomer wird in Nachrichten-Plug-Ins zum Zugriff auf die Kunden verwendet, für die Nachrichten erstellt werden. In Verzweigungs-Plug-Ins wird die Schnittstelle *IMHParticipant* verwendet, die von IMH-Customer erbt und diese erweitert.

Übergeordnete Schnittstelle

IMHCustomer erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer CustDomainID

Gibt die ID (eindeutige Kennung) der aktuellen Kundendomäne zurück. Schreibgeschützt.

string **CustomerID**

Ermöglicht den Zugriff auf die Zeichenfolge, die den Kunden repräsentiert. Schreibgeschützt.

string **Context**

Ermöglicht den Zugriff auf die Zeichenfolge, die den Kontext repräsentiert, in dem der Kunde auftritt. Schreibgeschützt.

variant **CustomValue**[*variant Value*Name]

Ermöglicht den Zugriff zum Lesen oder Festlegen eines benutzerdefinierten Wertes. Ein benutzerdefinierter Wert ist vom Datentyp „string“ (Zeichenfolge).

Benutzerdefinierte Werte sind Werte in einer für Dialogteilnehmer gespeicherten Name/Wert-Sammlung. Für jeden Teilnehmer steht eine beliebige Anzahl von Name/Wert-Paaren zur Verfügung.

Benutzerdefinierte Werte werden im Datenbankfeld DLG_PARTICIPANT.DP_CUSTOM_VALUES gespeichert.

Hinweis: Die Verwendung dieser Eigenschaft außerhalb des Kontexts eines Teilnehmers ist zwar nicht unmöglich, allerdings werden die Werte dann nicht in der Datenbank gespeichert.

bool **EditMode**

Gibt an, ob die Kundendaten geändert werden können.

Kundendaten können nach dem Aufrufen von *IMHCustomerContainer.NewCustomer( ) oder IMHCustomer.Edit( )* geändert werden. Siehe Beispiel weiter unten.

bool **HasDuplicate**

Gibt *true* (wahr) zurück, wenn potenzielle Duplikate des Kunden existieren. Die tatsächlich ausgeführte Überprüfung auf Duplikate muss in der Kundendomäneneinrichtung in Dialogue Admin konfiguriert werden.

string **MainDataGroupName**

Gibt den Namen der Hauptdatengruppe in der aktuellen Kundendomäne zurück. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **CamcelChanges**( )

Bricht Änderungen eines bestehenden oder neuen Kunden ab, die mit dem Aufrufen von *IMHCustomerContainer.NewCustomer( ) oder IMHCustomer.Edit( )* initiiert wurden.

IMHCustomerContainer **CheckForDuplicates**(*string* DataFields, *integer* MaxCount)

Überprüft, ob potenzielle Duplikate des Kunden existieren. Die Methode gibt ein Container-Objekt mit einer Liste potenzieller Duplikate zurück. Die tatsächlich ausgeführte Überprüfung auf Duplikate muss in der Kundendomäneneinrichtung in Dialogue Admin konfiguriert werden.

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern, die zum Initialisieren des Containers potenzieller Duplikate verwendet wird.

MaxCount ist die Maximalzahl der zurückgegebenen möglichen Duplikate. Setzen Sie *MaxCount* auf -1, um alle potentiellen Duplikate zu erhalten.

IMHAnswerForm **CreateAnswerForm**(*integer* QuestionnaireID, *string* ChannelTypeName, *variant* BroadcastID, *datetime* AnswerTimestamp*, datetime* ScanTimestamp, *string* ArchiveRef)

Erstellt ein neues Antwortformular in der Datenbank, und gibt ein Objekt zurück, das dieses Antwortformular repräsentiert.

QuestionnaireID ist die ID des Fragebogens.

ChannelTypeName ist der technische Name des Kommunikationskanals, über den die Antworten empfangen werden.

BroadcastID ist die ID der mit dem Antwortformular verbundenen Übertragung. Legen Sie *BroadcastID* auf *null* fest, wenn keine Übertragung verbunden werden soll.

AnswerTimestamp ist Datum und Zeitpunkt, zu dem der Kunde den Fragebogen beantwortet hat.

ScanTimestamp ist Datum und Zeitpunkt, zu dem die Antwortformulare eingescannt wurden. Legen Sie diesen Wert auf *null* fest, wenn dies nicht zutrifft.

ArchiveRef ist ein optionaler Wert.

void **CreateMergeFile***(string FileName, string DataFields, bool UseCharDelimiter, bool UseQuote, char CharDelimiter, char QuoteChar)*

„CreateMergeFile“ ist eine Hilfsfunktion, die Kundendaten in eine Textdatei exportiert.

FileName ist der vollständige Pfad und Name der Zieldatei.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern, die in der Datei enthalten sein sollen. Datengruppen, die mehrere Zeilen pro Kunden zurückgeben, werden nicht unterstützt.

UseCharDelimiter gibt an, ob die Spalten in der Datei mit einem Zeichen separiert werden sollen. Anderenfalls wird die Trennung mit Tabulatoren verwendet. *CharDelimiter* gibt das Trennzeichen an.

UseQuote gibt an, ob Feldwerte in Anführungszeichen gesetzt werden sollen. *QuoteChar* ist das einzelne Zeichen, das als Anführungszeichen verwendet werden soll.

Diese Methode wird vom Zusammenführungs-Plug-In von MS Word verwendet, um eine Datendatei zu erzeugen, bevor Word-Dokumente zusammengefügt werden. Diese Methode kann zudem zur Erstellung von Exportdateien verwendet werden.

IMHMessage CreateMessage( )

Erstellt ein leeres Nachrichtenobjekt, das sich auf *einen einzelnen* Kunden bezieht.

Um ein Nachrichtenobjekt zu erstellen, das sich auf eine Reihe von Kunden bezieht, verwenden Sie die *CreateMessage*-Methode des Containerobjekts (IMHCustomerContainer).

void **Delete**( )

Löscht den aktuellen Kunden.

Der Kunde wird aus allen Datenbanktabellen in Dialogue Database gelöscht, wie z. B. DLG_PARTICIPANT und QRY_ANSWER_FORM. Zusätzlich wird der Kunde entsprechend der Konfiguration in der Kundendomäne gelöscht. Demzufolge wird der Kunde möglicherweise aus anderen Datenquellen gelöscht, wie z. B. einer zugrunde liegenden Kundendatenbank.

void **DeleteDetailData**(*string* DataGroupName)

Löscht eine Zeile in einer (untergeordneten) Detail-Datengruppe des Kunden, zum Beispiel eine Adresszeile.

DataGroupName ist der Name der Datengruppe, in der die Zeile gelöscht wird.

Hinweis: *DeleteDetailData( )* kann sowohl von 1:1- als auch von 1:n-Gruppen aufgerufen werden.

void **DeleteLogin**( )

Löscht eine Kennwortdefinition in der Datenbanktabelle CUST_LOGIN. Wenn die Einstellung der Kundendomäne die Verwendung autogenerierter Kennwörter vorsieht, wird das aktuelle Kennwort des Kunden gelöscht, und dieser erhält ein neues Kennwort bei der nächsten Gelegenheit, bei der ein Kennwort verlangt wird.

bool **DetailDataEof**(*string* DataGroupName)

Beim Traversieren von Daten in 1:n-Datengruppen in der Kundendomäne werden die *FirstDetailData-*, *NextDetailData-* und *DetailDataEof-*Methoden verwendet. *DetailDataEof* gibt *true* (wahr) zurück, wenn die letzte Zeile der aufgerufenen Datengruppe erreicht wurde.

DataGroupName ist der Name der aufgerufenen Datengruppe.

void **Edit**( )

Versetzt den Kunden in den Bearbeitungsmodus. Nach dem Aufrufen von *Edit( )* werden Kundendaten möglicherweise mithilfe von *SetFieldValue( )* geändert.

Hinweis: *Edit( )* kann nur aufgerufen werden, wenn die Öffnung des Kundencontainers mit einem Aufruf von Folgendem erfolgte: *OpenForUpdates( ).* Siehe Beispiel weiter unten.

bool **EvalExprBool**(*string* Expression)

Wertet das Ergebnis eines booleschen Ausdrucks für den aktuellen Kunden aus und gibt das Ergebnis zurück.

Expression ist der auszuwertende Ausdruck. Die Angabe eines nicht-booleschen Ausdrucks führt zu einer Fehlermeldung.

string **EvalExprString**(*string* Expression)

Wertet das Ergebnis eines Ausdrucks für den aktuellen Kunden aus und gibt das Ergebnis zurück. Das Ergebnis wird als Zeichenfolge zurückgegeben.

Expression ist der auszuwertende Ausdruck. Die Angabe eines Ausdrucks, der eine Datengruppe oder ein Array zurückgibt, führt zur Ausgabe einer Ausnahme.

variant **EvalExprToValue**(*string* Expression)

Wertet das Ergebnis eines Ausdrucks für den aktuellen Kunden aus und gibt das Ergebnis zurück.

Expression ist der auszuwertende Ausdruck. Die Angabe eines Ausdrucks, der eine Datengruppe oder ein Array zurückgibt, führt zur Ausgabe einer Ausnahme.

variant **FieldValue**(*string* FieldName)

Gibt den Wert eines in der aktuellen Kundendomäne definierten Datenfeldes zurück. Der zurückgegebene Datentyp hängt vom Datentyp des aufgerufenen Feldes ab.

FieldName ist der Name des Feldes, auf das zugegriffen wird. Verwenden Sie den Gruppennamen als Präfix, um auf Felder in anderen Gruppen als der Hauptdatengruppe zuzugreifen. <Gruppenname >. <Feldname>, z. B. „Adresse.Straße“.

void **FirstDetailData**(*string* DataGroupName)

Beim Traversieren von Daten in 1:n-Datengruppen in der Kundendomäne werden die *FirstDetailData-*, *NextDetailData-* und *DetailDataEof*-Methoden verwendet. *FirstDetailData* legt die erste Zeile in der Gruppe als die aktuelle Zeile fest.

DataGroupName ist der Name der aufgerufenen Datengruppe.

IMHAnswerForm GetAnswerForm(*integer* QuestionnaireID)

Gibt ein Objekt zurück, das ein mit dem Kunden verbundenes Antwortformular repräsentiert.

QuestionnaireID ist die ID des Fragebogens.

Hinweis: Das Antwortformular muss mit dem Kunden und dem Kontext (falls verwendet) des aktuellen Kunden oder Teilnehmers übereinstimmen. Wenn kein übereinstimmendes Antwortformular existiert, wird *null* zurückgegeben.

string **GetCustomerAsXML**( )

Gibt ein XML-Dokument mit Kundendaten zurück.

Die zurückgegebenen Kundendaten hängen vom Inhalt der *DataFields*-Zeichenfolge in einem vorherigen Aufruf ab, der zur *Open*-Methode des Besitzers des Containerobjekts (IMHCustomerContainer) im Kundenobjekt gehört.

IMHDataField GetFieldInfo(*string* FieldName)

Ermöglicht den Zugriff auf ein Objekt mit Informationen über ein Kundendomänenfeld.

string **GetLoginID**( )

Gibt die Anmelde-ID des aktuellen Kunden zurück. Die Anmelde-ID ist in der Kundendomänenkonfiguration in Dialogue Admin definiert.

string **GetLoginPassword**(*bool* Regenerate)

Gibt das Kennwort des aktuellen Kunden zurück. Das Kennwort ist in der Kundendomänenkonfiguration in Dialogue Admin definiert.

Wenn festgelegt wurde, dass das Kennwort von Dialogue Server autogeneriert wird, kann es erneuert werden (ein neues Kennwort wird generiert), indem *Regenerate* auf *true* (wahr) festgelegt wird.

IMHUnmergedMessage GetMessageTemplate(*integer* BaseMessageID)

Gibt eine Nachrichtenvorlage entsprechend einer Nachrichtenvorlage in Visual Dialogue zurück.

BaseMessageID ist die eindeutige ID der Nachrichtenvorlage. Diese ID findet man in der Datenbanktabelle DOC_BASE_MESSAGE in der Spalte DBM_ID.

IMHUnmergedMessage GetUnmergedMessage(*string* MessageTypeName)

Gibt ein Vorlagenobjekt eines angegebenen, in Dialogue Admin definierten Nachrichtentyps zurück. Beachten Sie, dass das zurückgegebene Objekt nicht auf einer speziellen Mastervorlage oder Nachrichtenvorlage basiert.

MessageTypeName ist der technische Name des Nachrichtentyps, z. B. EMAIL_HTML.

bool **HasInclompleteResponse**(*integer* QuestionnaireID, *integer* LayoutID, *integer* PageIndex)

Gibt *true* (wahr) zurück, wenn der Kunde angefangen hat, den angegebenen Fragebogen zu beantworten, die Beantwortung aber nicht abgeschlossen hat. *HasIncompleteResponse( )* kann nur vorliegen, wenn *Antwortnachverfolgung* aktiviert ist.

LayoutID ist die eindeutige ID des verwendeten Fragebogenlayouts.

PageIndex ist die Nummer der eingegebenen Seite (d. h. der dem Antwortenden angezeigten Seite) in der unvollständigen Antwort. *PageIndex* beginnt mit 1 auf der ersten Seite des Layouts.

bool **HasOpenedEmail**(*integer* TemplateID)

Gibt *true* (wahr) zurück, wenn der Kunde eine E-Mail mit aktivierter *E-Mail-Nachverfolgung* geöffnet hat.

TemplateID ist die eindeutige ID einer Nachrichtenvorlage, die in Visual Dialogue entworfen wurde (entsprechend einem Wert in der Datenbankspalte DOC_BASE_MESSAGE.DBM_ID).

Hinweis: *HasOpenedEmail( )* wird mit dem E-Mail-Nachverfolgungsmechanismus in Dialogue Server verwendet. Wenn einem Kunden eine HTML-E-Mail-Nachricht gesendet wird und „E-Mail-Nachverfolgung“ in dieser Nachricht aktiviert ist, wird das System nachverfolgen, wann der Kunde diese E-Mail öffnet.

bool **HasOpenedLink**(*integer* TemplateID, *string* LinkName)

Gibt *true* (wahr) zurück, wenn ein Kunde einen angegebenen Link geöffnet hat und *Verknüpfungsnachverfolgung* aktiviert ist.

„TemplateID“ ist die eindeutige ID einer in Visual Dialogue entworfenen Nachrichtenvorlage (entsprechend eines Werts in der Datenbankspalte DOC_BASE_MESSAGE.DBM_ID). Wenn *TemplateID* auf 0 festgelegt wurde, wird von der Funktion die Überprüfung über alle Nachrichten hinweg ausgeführt (unabhängig von der Vorlage).

LinkName ist der Name der nachverfolgten Verknüpfung, wie diese beim Entwurf der Vorlage im Nachrichten-Designer bzw. beim Aufrufen der Ausdrucksfunktion „TrackURL“ angegeben wurde.

Hinweis: *HasOpenedLink( )* wird mit dem Verknüpfungsnachverfolgungsmechanismus in Dialogue Server verwendet. Eine E-Mail-Nachricht kann Verknüpfungen (URLs) enthalten, für welche die Verknüpfungsnachverfolgung aktiviert ist. Das System verfolgt, wenn Kunden diese Verknüpfungen in den E-Mails öffnen, die sie erhalten haben.

bool **HasOpenedLinkEx**(*integer* TemplateID, *string* Url)

HasOpenedLinkEx( ) bewirkt das Gleiche wie *HasOpenLink( )*, außer dass einer der Parameter eine URL anstelle des Namens der nachverfolgten Verknüpfung ist.

Url ist die Adresse (URL) der nachverfolgten Verknüpfung.

void **LogBehaviour** *(integer* BehaviourTypeID, *variant* TreatmentID, *variant* OfferID, *variant* ProductCode, *variant* TreatmentID, *integer* ResponseIndicator, *variant* ResponseValue, *variant* Origin, *datetime* ActedTimestamp)

Protokolliert ein Kundenverhalten eines angegebenen Verhaltenstyps. Die protokollierten Daten werden asynchron zum PDW (Portrait Data Warehouse) übertragen und zur Speicherung von Antworten auf Kampagnen und Kampagnenaktivitäten verwendet.

BehaviourTypeID ist die eindeutige ID des Verhaltenstyps gemäß der Definition im PSR (Portrait Shared Repository). Dieser Parameter ist obligatorisch.

TreatmentID ist die eindeutige ID der Behandlung in der Kampagnenaktivität gemäß der Definition im PSR. Dieser Wert kann auf „0“ (null) festgelegt werden.

OfferID ist die eindeutige ID des Angebots gemäß der Definition im PSR. Dieser Wert kann auf „0“ (null) festgelegt werden.

ProductCode ist ein benutzerdefinierter eindeutiger Produktcode (Zeichenfolge), um ein Verhalten mit einem bestimmten Produkt zu verbinden. Dieser Wert kann auf „0“ (null) festgelegt werden.

ResponseIndicator ist ein Ganzzahlwert, der anzeigt, ob eine aus dieser Verhaltensaufzeichnung gespeicherte Antwort negativ, neutral oder positiv ausgefallen ist. Die folgenden Werte sind gültig:
1 = positiv (Standard), 0 = neutral, –1 = negativ.

ResponseValue ist ein optionaler Wert, der den Wert oder die Bewertung einer gespeicherten Antwort in Bezug auf dieses Verhalten anzeigt. *ReponseValue* kann auf „0“ (null) festgelegt werden.

Origin ist eine benutzerdefinierte Zeichenfolge, die protokolliert wird. Dieser Wert wird nicht vom System verwendet, sondern ist ein optionaler Wert, der von einem Benutzer verwendet werden kann. Dieser Wert kann auf „0“ (null) festgelegt werden.

ActedTimestamp ist der tatsächliche Zeitpunkt, zu dem das Kundenverhalten stattgefunden hat.

Hinweis: Diese Methode dient zur Integration mit Portrait HQ und Portrait PSR.

void **LogOffer(** *integer* OfferID, *datetime* TreatedTimestamp)

Protokolliert, dass ein Kunde mit einem Angebot behandelt wurde. Die protokollierten Daten werden asynchron zum PDW (Portrait Data Warehouse) übertragen und zur Speicherung von Behandlungen und Angeboten verwendet.

OfferID ist die eindeutige ID des Angebots gemäß der Definition im PSR.

TreatedTimestamp ist der tatsächliche Zeitpunkt, zu dem der Kunde mit dem Angebot behandelt wurde.

Hinweis: Um diese Methode zur Protokollierung eines gültigen Behandlungsangebots-Protokolldatensatzes zu verwenden, muss die Dialogverzweigung, die diese Methode ausführt, mit der korrekten *TreatmentID* eingerichtet werden. Um sowohl *TreatmentID* als auch *OfferID* anzugeben, verwenden Sie stattdessen die *LogTreatmentOffer*-Methode.

Hinweis: Diese Methode dient zur Integration mit Portrait HQ und Portrait PSR.

void **LogTreatmentOffer(** *integer* TreatmentID, *variant* OfferID, *datetime* TreatedTimestamp)

Protokolliert, dass ein Kunde sowohl mit einem als auch ohne ein Angebot behandelt wurde. Die protokollierten Daten werden asynchron zum PDW (Portrait Data Warehouse) übertragen und zur Speicherung von Behandlungen und Angeboten verwendet.

TreatmentID ist die eindeutige ID der Behandlung in einer Kampagnenaktivität gemäß der Definition im PSR.

OfferID ist die eindeutige ID des Angebots gemäß der Definition im PSR. Dieser Wert sollte auf *null* festgelegt werden, wenn die Behandlung kein Angebot beinhaltet.

TreatedTimestamp ist der tatsächliche Zeitpunkt, zu dem der Kunde mit dem Angebot behandelt wurde.

Hinweis: Wenn die Behandlung mehrere Angebote enthält, muss die *LogTreatmentOffer*-Methode für jedes dieser Angebote einmal aufgerufen werden.

Hinweis: Diese Methode dient zur Integration mit Portrait HQ und Portrait PSR.

string **MergeControlParams**(*string* ControlParams)

Gibt eine Zeichenfolge mit Steuerungsparametern und zusammengeführten Kundendaten zurück. Diese Methode wird von Nachrichten-Plug-Ins verwendet, um zusammengeführte Felder mit Kundendatenwerten in den Steuerparametern der Vorlagen oder der Basisnachrichten zu ersetzen.

ControlParams ist die Zeichenfolge, welche die zusammenzuführenden Steuerparameter enthält.

void **NewDetailData**(*string* DataGroupName)

Fügt eine Zeile in einer (untergeordneten) Detail-Datengruppe des Kunden hinzu, zum Beispiel eine Adresszeile.

DataGroupName ist der Name der Datengruppe, in der die Zeile hinzugefügt wird.

Hinweis: *NewDetailData( )* sollte ausschließlich für 1:n-Gruppen aufgerufen werden. Siehe Beispiel weiter unten.

bool **NextDetailData**(*string* DataGroupName)

Beim Traversieren von Daten in 1:n-Datengruppen in der Kundendomäne werden die *FirstDetailData-*, *NextDetailData*- und *DetailDataEof*-Methoden verwendet. *NextDetailData* ruft die nächste Zeile in der aufgerufenen Datengruppe auf. Es wird *true* (wahr) zurückgegeben, wenn die letzte Zeile erreicht wurde und der Cursor nicht zur nächsten Zeile verschoben werden kann, anderenfalls wird *false* (falsch) zurückgegeben.

DataGroupName ist der Name der aufgerufenen Datengruppe.

int64 **PostActivity**(*string* ActivityTypeName, *string* ChannelTypeName, *string* Direction*, string* Description, *string* Note, *datetime* Timestamp)

Sendet eine Aktivität und gibt die eindeutige ID der neuen Aktivität zurück**.** Aktivitäten sind Kundeninteraktionen.

ActivityTypeName ist der technische Name des in Dialogue Admin definierten Aktivitätstyps.

ChannelTypeName ist der technische Name des in Dialogue Admin definierten Kanaltyps. Der Kanaltyp beschreibt die Art und Weise der stattgefundenen Kommunikation in der Aktivität oder Interaktion.

Direction beschreibt, ob die Aktivität vom Kunden oder von „uns“ bzw. „dem System“ initiiert wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN": initiiert vom Kunden.
- Direction = "OUT": initiiert von „uns“ bzw. „dem System“.

Description ist ein kurzes Beschreibungsfeld, während es sich bei *Note* um eine unbegrenzt lange Beschreibung handelt.

Timestamp ist der Zeitpunkt, zu dem die Aktivität stattgefunden hat.

void **PostChanges**( )

Speichert die Änderungen eines bestehenden oder eines neuen Kunden in den zugrunde liegenden Datenquellen entsprechend der Konfiguration der Kundendomäne.

Hinweis: Änderungen in Bezug auf Kunden werden mit Aufrufen von *IMHCustomerContainer.NewCustomer( ) oder IMHCustomer.Edit( )* initiiert. Siehe Beispiel weiter unten.

void **PostEvent**(*string* EventTypeName*, string* Description)

Postet ein Systemereignis. Systemereignisse werden in Dialogue Admin definiert. Dialogue Server kann so eingerichtet werden, dass auf unterschiedliche Systemereignisse mit unterschiedlichen Aktivitäten reagiert wird. Beispielsweise kann ein Vorgang in einem oder mehreren Dialogfeldern ausgeführt werden.

EventTypeName ist die technische Bezeichnung des in Dialogue Admin definierten Ereignistyps.

Description ist die optionale Beschreibung eines Ereignisses.

int64 **PostMessage**(**IMHUnmergedMessage** UnmergedMessage, **IMHMessage** Message, *bool* UseOutbox)

PostMessage speichert eine einzelne Nachricht (oder deren Dateinamen) in Dialogue Database. Dadurch wird diese zudem in den *Postausgang* oder die Sendewarteschlange gelegt. Die Sendewarteschlange ist die Warteschlange, von der aus der MH-Nachrichtenversanddienst elektronische Nachrichten wie E-Mails oder SMS-Nachrichten verschickt. Die ID der neuen Nachricht wird zurückgegeben.

UnmergedMessage ist ein Nachrichtenvorlagenobjekt, das den zu speichernden Nachrichtentyp beschreibt. Die *UnmergedMessage*-Objekte können mithilfe der *GetUnmergedMessage*- oder *GetMessageTemplate*-Methoden erstellt werden.

Message ist ein einzelnes Nachrichtenobjekt, das üblicherweise mithilfe der *ProduceMessage*-Methode erstellt wird.

UseOutbox gibt an, ob Nachrichten vor dem Senden in den Postausgang verschoben werden.

int64 **PostTask**(*string* ActivityTypeName, *string* ChannelTypeName, *string* Direction*, string* Description, *string* Note, *datetime* Timestamp, *datetime* FollowUpDateTime, *integer* TaskWorkGroupID, *string* FollowUpUserName)

Sendet eine Aufgabe und gibt die eindeutige ID der neuen Aktivität zurück. Aufgaben sind Aktivitäten, die innerhalb einer bestimmten Zeitdauer nachverfolgt werden.

ActivityTypeName ist der technische Name des in Dialogue Admin definierten Aktivitätstyps.

ChannelTypeName ist der technische Name des in Dialogue Admin definierten Kanaltyps. Der Kanaltyp beschreibt die Art und Weise der stattgefundenen Kommunikation in der Aktivität oder Interaktion.

Direction beschreibt, ob die Aktivität vom Kunden oder von „uns“ bzw. „dem System“ initiiert wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN": initiiert vom Kunden.
- Direction = "OUT": initiiert von „uns“ bzw. „dem System“.

Description ist ein kurzes Beschreibungsfeld, während es sich bei *Note* um eine unbegrenzt lange Beschreibung handelt.

Timestamp ist der Zeitpunkt, zu dem die Aktivität stattgefunden hat.

FollowUpDateTime ist der Wert für Datum und Uhrzeit, mit dem der Abgabetermin für die Nachverfolgung der Aufgabe festgelegt wird. *TaskWorkGroup* ist ein optionaler Wert, der sich auf die für die Nachverfolgung der Aufgabe verantwortliche Arbeitsgruppe bezieht. Setzen Sie diesen Parameter auf *null*, wenn keine Arbeitsgruppe angegeben werden soll.

FollowUpUserName ist der Name des für die Nachverfolgung der Aufgabe verantwortlichen Benutzers. Legen Sie diesen Parameter auf eine leere Zeichenfolge fest, wenn Sie keinen verantwortlichen Benutzer angeben möchten.

IMHMessage ProduceMessage(**IMHUnmergedMessage** UnmergedMessage)

ProduceMessage weist Dialogue Server an, eine Nachricht (z. B. E-Mail-Nachrichten) für den Kunden zu erstellen. Die Nachricht basiert auf einer in *UnmergedMessage* vorgegebenen Vorlage oder Basisnachricht.

Ein Nachrichtenobjekt wird zurückgegeben.

void **RemoveCategory**(*string* CategoryName)

Entfernt den Kunden aus einer Kategorie. Dies bedeutet, dass die Mitgliedschaft in der Kategorie gelöscht wird.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

void **RemoveCategoryValue**(*string* CategoryName, *string* Value)

Entfernt den Wert des Kunden in einer Kategorie. Die Kategorie muss den Typ *Kategorie mit Werten* haben.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Value den zu entfernenden String-Wert, der in der Kategorie Setup in Dialogue Admin definiert wird.

void **SetCategory**(*string* CategoryName)

Fügt den Kunden zu einer Kategorie hinzu. Dies bedeutet, dass eine Kategoriemitgliedschaft erstellt wird.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

void **SetCategoryScore**(*string* CategoryName, *float* Score)

Fügt den Kunden zu einer Kategorie mit Bewertungswert hinzu. Dies bedeutet, dass eine Kategoriemitgliedschaft erstellt oder der Bewertungswert einer bestehenden Mitgliedschaft aktualisiert wird.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Score ist der Bewertungswert zwischen den Minimal- und Maximalwerten, die in der Kategorieeinrichtung in Dialogue Admin definiert sind.

void **SetCategoryValue**(*string* CategoryName, *string Value*)

Fügt den Kunden zu einer Kategorie mit einem Wert hinzu. Dies bedeutet, dass eine Kategorienmitgliedschaft erstellt oder ein Wert zu einer bestehenden Kategorie hinzugefügt wird.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Value bezeichnet den einzurichtenden String-Wert, der in der Kategorie Setup in Dialogue Admin definiert wird.

void **SetFieldValue**(*string* FieldName, *variant* Value)

Legt den Wert eines Datenfeldes fest, das in der Kundendomäne definiert wurde.

FieldName ist der Name des Feldes, auf das zugegriffen wird. Verwenden Sie den Gruppennamen als Präfix, um auf Felder in anderen Gruppen als der Hauptdatengruppe zuzugreifen. <Gruppenname >. <Feldname>, z. B. „Adresse.Straße“.

Value ist der neue Wert des Feldes.

Hinweis: Der Kunde muss im Bearbeitungs- oder Einfügemodus sein, bevor Daten durch Verwendung des folgenden Befehls geändert werden können: *SetFieldValue( ).* Der Einfügemodus wird durch das Aufrufen von *IMHCustomerContainer.NewCustomer( )* initiiert. Der Bearbeitungsmodus wird durch das Aufrufen von *IMHCustomer.Edit( )* initiiert. Siehe Beispiel weiter unten.

void **SetLoginPassword**(*string* NewPassword)

Legt ein Kundenkennwort fest. Der neue Kennwortwert wird in der Datenbanktabelle CUST_LOGIN gespeichert.

NewPassword ist das neue Kennwort des Kunden.

Hinweis: Es ist nur möglich das Kennwort festzulegen, wenn Kennwörter so eingerichtet sind, dass sie durch Dialogue Server automatisch generiert werden.

Beispiele

Das folgende Beispiel zeigt, wie man auf Kundeninformationen zugreift:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   //Open the container
   Participants.Open("mh_customer_id;Address");

   while (! Participants.MemberEof) {

     //Do something with the customer
     var Street = Participants.Customer.FieldValue("Address.StreetName");
     Participants.Customer.SetCategory("NEWSLETTER");

     ..............
     //Move to next participants
     Participants.NextMember();
   }
   ..............

            }
```

Das folgende Beispiel zeigt, wie man Kundendaten in der Dialogoperation bearbeitet:

```
function ExecuteBranch(BranchInfo, Participants)
{
//Open the customer container in editable mode
Participants.OpenForUpdates("contact;company;adresser");

while ((!Participants.MemberEof) && (i < Participants.Count))
{
//Start edit the customer
Participants.Participant.Edit();

    //Update last name of customer (main data group)
Participants.Participant.SetFieldValue("Etternavn", "Peterson");
    //Delete old addresses
Participants.Participant.FirstDetailData("Adresser");
while (!Participants.Participant.DetailDataEof("Adresser"))
{
Participants.Participant.DeleteDetailData("Adresser");
}

//Create a new address (one2many group) (address type 2)
Participants.Participant.NewDetailData("Adresser");
Participants.Participant.SetFieldValue("Adresser.pa_at_id", "2");
Participants.Participant.SetFieldValue("Adresser.Gatenavn", "Maridalsveien");
Participants.Participant.SetFieldValue("Adresser.Gatenummer", "87");
Participants.Participant.SetFieldValue("Adresser.Bokstav", "C");


    //Create a new address (one2many group) (address type 3)
Participants.Participant.NewDetailData("Adresser");
Participants.Participant.SetFieldValue("Adresser.pa_at_id", "3");
Participants.Participant.SetFieldValue("Adresser.Gatenavn", "River Street");
Participants.Participant.SetFieldValue("Adresser.Gatenummer", "7");

//Delete company relation (one2one group) (not neccessary when changing the
 relation)
//Participants.Participant.DeleteDetailData("Company");

//Add a new company relation
Participants.Participant.SetFieldValue("Company.CompanyID", 1919);

//Save changes to customer to the underlying data source as defined in the
customer domain
Participants.Participant.PostChanges();

    //Move participant to the next group in the dialog
Participants.Participant.Accept();

    Participants.NextMember();
}
}
}
```

IMHCustomerContainer-Schnittstelle

Beschreibung

IMHCustomerContainer bietet Zugriff auf eine Reihe oder Auswahl von Kunden.

Anwendung

„IMHCustomerContainer“ wird in Nachrichten-Plug-Ins verwendet, um Zugriff auf die Kunden zu gewähren, für die Nachrichten erstellt werden. In Verzweigungs-Plug-Ins wird die Schnittstelle *IMHParticipantContainer* verwendet, welche von „IMHCustomerContainer“ erbt und diese erweitert.

Übergeordnete Schnittstelle

„IMHCustomerContainer“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **AllDataFields**

Schreibgeschützte Eigenschaft, die Zugriff auf alle verfügbaren Datenfelder in der aktuellen Kundendomäne bietet. Die Datenfelder werden als eine durch Semikola getrennte Zeichenfolge aufgelistet.

integer **Count**

Gibt die Anzahl der Kunden im Container zurück. Schreibgeschützt.

integer **CustDomainID**

Gibt die ID (eindeutige Kennung) der aktuellen Kundendomäne zurück. Schreibgeschützt.

IMHCustomer Customer

Gibt einen Bezug zu einem Objekt zurück, das den aktuellen Kunden darstellt. Schreibgeschützt.

string **DataFields**

Schreibgeschützte Eigenschaft, die Zugriff auf die Kundendatenfelder bietet, die im Container enthalten sind. Dies sind dieselben Felder, die bei einem vorherigen Aufruf der Methode *Open* angegeben wurden. Die Datenfelder werden als eine durch Semikola getrennte Zeichenfolge aufgelistet.

int64 **InternalID**

Schreibgeschützte Eigenschaft, die Zugriff auf einen internen Wert bietet, der während der aktuellen Transaktion verwendet wird.

Hinweis: Dieser Wert wird normalerweise nicht in Plug-Ins verwendet. Allerdings muss dieser Wert beim Verwenden erweiterter Methoden wie *IMHParticipantContainer.AcceptBySQL(...)* und *IMHParticipantContainer.AcceptBySQLRaw(...)* genutzt werden.

integer **MaxCount**

Gibt die maximale Anzahl von Kunden an, die im Container enthalten sind, wie bei der Initialisierung des Containers angegeben. Schreibgeschützt.

string **MainDataGroupName**

Gibt den Namen der Hauptdatengruppe in der aktuellen Kundendomäne zurück. Schreibgeschützt.

bool **MemberEof**

Zeigt an, ob der letzte Kunde im Container erreicht wird. Beachten Sie, dass „MemberEof“ erst auf *true* gesetzt wird, wenn *NextMember* aufgerufen wird, ohne dass der aktuelle Kunden-Cursor zum nächsten Kunden bewegt werden kann. Schreibgeschützt.

bool **Opened**

Zeigt an, ob der Container zum Traversieren der Kunden geöffnet wurde. Schreibgeschützt.

bool **Updatable**

Zeigt an, ob Kundendaten geändert und neuen Kunden hinzugefügt werden können.

Updatable zeigt *true* an, nachdem der Kundencontainer durch das Aufrufen von *OpenForUpdates( )* initiiert wurde. Siehe Beispiel weiter unten.

Methoden

Die folgenden Methoden werden unterstützt:

void **CreateMergeFile***(string FileName, string DataFields, bool UseCharDelimiter, bool UseQuote, char CharDelimiter, char QuoteChar)*

„CreateMergeFile“ ist eine Hilfsfunktion, die Kundendaten in eine Textdatei exportiert.

FileName ist der vollständige Pfad und Name der Zieldatei.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern, die in der Datei enthalten sein sollen. Datengruppen, die mehrere Zeilen pro Kunden zurückgeben, werden nicht unterstützt.

UseCharDelimiter gibt an, ob die Spalten in der Datei mit einem Zeichen separiert werden sollen. Anderenfalls wird die Trennung mit Tabulatoren verwendet. *CharDelimiter* gibt das Trennzeichen an.

UseQuote gibt an, ob Feldwerte in Anführungszeichen gesetzt werden sollen. *QuoteChar* ist das einzelne Zeichen, das als Anführungszeichen verwendet werden soll.

Diese Methode wird vom Zusammenführungs-Plug-In von MS Word verwendet, um eine Datendatei zu erzeugen, bevor Word-Dokumente zusammengefügt werden. Die Methode kann auch verwendet werden, um Exportdateien mit enthaltenen Kundendaten zu erstellen.

IMHMessage CreateMessage( )

Erstellt ein leeres Nachrichtenobjekt, das mit *allen* Kunden im Container verbunden ist. Diese Methode ist beispielsweise geeignet, wenn eine Exportdatei erstellt wird, die mehrere Kunden enthält.

Verwenden Sie die Methode *CreateMessage* des Kundenobjekts („IMHCustomer“), um ein Nachrichtenobjekt zu erstellen, das mit einem einzelnen Kunden verbunden ist.

IMHMessageBundle CreateMessageBundle(**IMHUnmergedMessage** UnmergedMessage)

Erstellt ein leeres Nachrichtenpaketobjekt basierend auf einer Vorlage oder Basisnachricht, die in *UnMergedMessage* angegeben wird.

Die Methode „CreateMessageBundle“ wird meistens in Nachrichten-Plug-Ins verwendet. Allerdings kann sie auch in Verzweigungs-Plug-Ins eingesetzt werden.

void **FirstMember**( )

Legt den ersten Kunden im Container als den aktuellen fest.

string **GetCustomersAsXML**( )

Gibt eine Zeichenfolge zurück, die ein XML-Dokument mit Daten aller Kunden im Container enthält. Die zurückgegebenen Kundendaten sind abhängig von der Zeichenfolge *DataFields* in einem vorherigen Aufruf der Methode *Open*.

IMHUnmergedMessage GetMessageTemplate(*integer* BaseMessageID)

Gibt eine Nachrichtenvorlage entsprechend einer Nachrichtenvorlage in Visual Dialogue zurück.

BaseMessageID ist die eindeutige ID der Nachrichtenvorlage. Diese ID findet man in der Datenbanktabelle DOC_BASE_MESSAGE in der Spalte DBM_ID.

IMHUnmergedMessage GetUnmergedMessage(*string* MessageTypeName)

Gibt ein Vorlagenobjekt eines angegebenen, in Dialogue Admin definierten Nachrichtentyps zurück. Beachten Sie, dass das zurückgegebene Objekt nicht auf einer speziellen Mastervorlage oder Nachrichtenvorlage basiert.

MessageTypeName ist der technische Name des Nachrichtentyps, z. B. EMAIL_HTML.

bool **NextMember** ( )

Legt den nächsten Kunden im Container als aktuellen fest. Gibt *true* zurück, wenn der aktuelle Kunde *nicht* geändert wurde, weil der letzte Kunde erreicht wurde.

void **NewCustomer**( )

Startet den Prozess des Einfügens eines neuen Kunden in den Container.

Hinweis: *NewCustomer( )* kann nur aufgerufen werden, wenn der Kundencontainer durch Aufrufen von *OpenForUpdates( )* geöffnet wurde.

void **Open**(*string* DataFields)

Öffnet den Container zum Traversieren seiner Kunden.

DataFields ist eine durch Semikola getrennte Liste der Kundendatengruppen und -felder (aus der Kundendomänendefinition) zum Zwischenspeichern im Container. Auf andere Felder kann ebenfalls zugegriffen werden. Das Angeben von „DataFields“ optimiert jedoch die Datenübertragung in Dialogue Server.

Hinweis: *Open( )* oder *OpenForUpdates( )* muss vor dem Zugriff auf die Eigenschaft *Customer* aufgerufen werden.

void **OpenForUpdates**(*string* DataGroups)

Öffnet den Container zum Traversieren seiner Kunden im aktualisierbaren Modus. Aktualisierbarer Modus bedeutet, dass Kundendaten bearbeitet und neue Kunden eingefügt werden können.

DataGroups ist eine durch Semikola getrennte Liste der Kundendatengruppen (aus der Kundendatendefinition), die aktualisierbar sind, wenn Sie mit dem Kundencontainer arbeiten. Siehe Beispiel weiter unten.

Hinweis: *Open( )* oder *OpenForUpdates( )* muss vor dem Zugriff auf die Eigenschaft *Customer* aufgerufen werden.

void **OpenSorted**(*string* DataFields, *string* SortFields)

Bewirkt dasselbe wie *Open(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine durch Semikola getrennte Liste der Kundendomänenfelder, die verwendet wird, um die Kunden im Container zu sortieren.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

void **PostMessages**(**IMHMessageBundle** Messages, *bool* UseOutbox)

PostMessages speichert die erstellten Nachrichten (oder ihre Dateinamen) in Dialogue Database. Dadurch wird diese zudem in den *Postausgang* oder die Sendewarteschlange gelegt. Die Sendewarteschlange ist die Warteschlange, von der aus der MH-Nachrichtenversanddienst elektronische Nachrichten wie E-Mails oder SMS-Nachrichten verschickt.

Messages ist ein Nachrichtenpaket, das gewöhnlich durch Verwendung der Methode *ProduceMessages* erstellt wird.

UseOutbox gibt an, ob Nachrichten vor dem Senden in den Postausgang verschoben werden.

integer **PostSavedMessages**( )

PostSavedMessages( ) leitet Nachrichten, die zuvor in der aktuellen Transaktion unter Verwendung von SaveMessages(...) gespeichert wurden, an den *Postausgang* oder die *Sendewarteschlange*.

IMHMessageBundle **ProduceMessages**(**IMHUnmergedMessage** UnMergedMessage)

ProduceMessages weist Dialogue Server an, für alle Kunden im Container Nachrichten (z. B. E-Mail-Nachrichten) zu erstellen. Die Nachricht basiert auf einer in *UnmergedMessage* vorgegebenen Vorlage oder Basisnachricht.

Ein Nachrichtenpaket wird zurückgegeben. Dieses Objekt ist ein Container aller erstellten Nachrichten, und kann an die Methode *PostMessages* weitergeleitet werden, welche die Nachrichten in den *Postausgang* oder die *Sendewarteschlange* verschiebt.

IMHMessageBundle **ProduceMessagesEx**(**IMHUnmergedMessage** UnMergedMessage, *integer* MaxBundleSize)

Bewirkt dasselbe wie ProduceMessages(...). Allerdings ist die Anzahl der zurückgegebenen Nachrichten im Paket begrenzt auf die angegebene Anzahl.

MaxBundleSize ist die maximale Anzahl der Nachrichten, die erstellt werden.

Hinweis: Diese Methode wurde eingeführt, um die Speichernutzung zu reduzieren, wenn eine große Anzahl von E-Mails erstellt wird. Nicht alle Nachrichtentypen und -Plug-Ins unterstützen diese Methode.

void **Reset( )**

Schließt den Kundencontainer und deinitialisiert ihn. Nach dem Aufrufen von *Reset( )* kann der Container initialisiert und wieder geöffnet werden.

Hinweis: Das Aufrufen von *Reset( )* in einer Dialogoperation schließt den Container. Es ist möglich, ihn zu reinitialisieren und wieder zu öffnen. Allerdings werden beliebige, mit Teilnehmern durchgeführte Aktionen nicht rückgängig gemacht, z. B. wird ein bereits in die Empfängergruppe verschobener Teilnehmer nicht zurückverschoben.

void **SaveMessages**(**IMHMessageBundle** Messages, *bool* UseOutbox)

„SaveMessages“ bewirkt dasselbe wie „ProduceMessages“. Allerdings werden die Nachrichten nicht zum *Postausgang* oder der *Sendewarteschlange* geleitet. Ein Aufruf von *PostSavedMessage(...)* wird an einem Punkt innerhalb der Transaktion nach *SaveMessage(...)* benötigt. *PostSavedMessages(...)* leitet die Nachrichten dann zum Postausgang oder der Sendewarteschlange. Nach mehreren Aufrufen von *SaveMessages(...)* muss *PostSavedMessages(...)* nur einmal aufgerufen werden.

Beispiele

Das folgende Beispiel zeigt, wie man Kunden traversiert:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   //Open the container
   Participants.Open("mh_customer_id;Address");

   while (! Participants.MemberEof) {

     //Do something with the participant / customer

     //Move to next participants
     Participants.NextMember();
   }
   ..............

}
```

Das folgende Beispiel zeigt, wie man Kundendaten in der Dialogoperation bearbeitet:

```
function ExecuteBranch(BranchInfo, Participants)
{
//Open the customer container in editable mode
Participants.OpenForUpdates("contact;company;adresser");

while ((!Participants.MemberEof) && (i < Participants.Count))
{
```

```
//Start edit the customer
Participants.Participant.Edit();

    //Update last name of customer (main data group)
Participants.Participant.SetFieldValue("Etternavn", "Peterson");
    //Delete
 old addresses
Participants.Participant.FirstDetailData("Adresser");
while (!Participants.Participant.DetailDataEof("Adresser"))
{
Participants.Participant.DeleteDetailData("Adresser");
}

//Create a new address (one2many group) (address type 2)
Participants.Participant.NewDetailData("Adresser");
Participants.Participant.SetFieldValue("Adresser.pa_at_id", "2");
Participants.Participant.SetFieldValue("Adresser.Gatenavn", "Maridalsveien");
Participants.Participant.SetFieldValue("Adresser.Gatenummer", "87");
Participants.Participant.SetFieldValue("Adresser.Bokstav", "C");

   //Create a new address (one2many group) (address type 3)
Participants.Participant.NewDetailData("Adresser");
Participants.Participant.SetFieldValue("Adresser.pa_at_id", "3");
Participants.Participant.SetFieldValue("Adresser.Gatenavn", "River Street");
Participants.Participant.SetFieldValue("Adresser.Gatenummer", "7");

//Delete company relation (one2one group) (not neccessary when changing the
 relation)
//Participants.Participant.DeleteDetailData("Company");

//Add a new company relation
Participants.Participant.SetFieldValue("Company.CompanyID", 1919);

//Save changes to customer to the underlying data source as defined in the
customer domain
Participants.Participant.PostChanges();
    //Move participant to the next group in the dialog
Participants.Participant.Accept();

   Participants.NextMember();
}
}
}
```

IMHCustomerList-Schnittstelle

Beschreibung

IMHCustomerList ermöglicht den Zugriff auf Informationen über eine Kundenliste.

Anwendung

„IMHCustomerList“ wird in Plug-Ins für den Zugriff auf Kundenlisten verwendet.

Übergeordnete Schnittstelle

„IMHCustomerList“ erbt **IMHCustomPluginServices** von der Basisschnittstelle.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

*integer***Id**

Ermöglicht den Zugriff auf die eindeutige ID der Liste, die der Datenbankspalte <codeph>LIST.LST_ID</codeph> entspricht. Schreibgeschützt.

*string***Name**

Ermöglicht den Zugriff auf den Namen der Liste. Schreibgeschützt.

*bool***HasContext**

Gibt an, ob die Mitglieder der Liste durch eine Kunden-ID und einen Kontextwert oder nur durch eine Kunden-ID identifiziert werden.

Die Datenbanktabelle <codeph>LIST_MEMBER</codeph> enthält Informationen zu den Mitgliedern einer Liste. Wenn *HasContext* „true“ ist, wird der Kontextwert in der Spalte <codeph>LIST_MEMBER.LME_CONTEXT</codeph> gespeichert.

HasContext ist schreibgeschützt.

*bool***IsExplorer**

Gibt an, ob die Liste von Portrait Explorer erstellt wurde.

Hinweis: In PD 6.0 SP1 unterstützt nur Portrait Explorer das Erstellen von Listen, die in Portrait Dialogue verwendet werden können. Daher ist *IsExplorer* immer auf **true** gesetzt, sofern die Liste nicht manuell direkt in der Datenbank erstellt wurde.

Methoden

Die folgenden Methoden werden unterstützt:

Beispiele

Es sind keine Beispiele verfügbar.

IMHCustomerSortFieldList-Schnittstelle

Beschreibung

IMHCustomerSortFieldList ermöglicht den Zugriff auf eine Liste mit Kundenfeldern zum Sortieren.

Anwendung

IMHCustomerSortFieldList wird in Plug-Ins zum Zugriff auf Kundenfelder verwendet, um einen Kundencontainer zu sortieren.

Übergeordnete Schnittstelle

IMHCustomerSortFieldList erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **IsEmpty**

Gibt an, ob die Liste leer ist oder nicht. Schreibgeschützt.

string **ListAsString**

Ermöglicht den Zugriff auf eine durch Semikola getrennte Liste von Sortierfeldern. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHCustomPluginServices-Schnittstelle

Beschreibung

Eine Basisschnittstelle von der andere Schnittstellen erben.

Anwendung

Enthält die Basiseigenschaften und -methoden.

Eigenschaften

Die folgenden Eigenschaften sind verfügbar:

string **ObjectName**

Gibt den internen Namen des Objekts zurück, das die Schnittstelle implementiert. Schreibgeschützt.

Methoden

Es sind keine Methoden verfügbar.

Beispiele

Es stehen keine Beispiele zur Verfügung.

IMHDataField-Schnittstelle

Beschreibung

IMHDataField bietet Zugriff auf Feldwerte und -eigenschaften einer Kundendomäne.

Anwendung

„IMHDataField“ wird beispielsweise in Kundendaten-Plug-Ins verwendet, wenn Kundendaten aktualisiert werden.

Übergeordnete Schnittstelle

„IMHDataField“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ContentType**

Gibt den Inhaltstyp des Feldes zurück, wenn dieser zugewiesen wurde. Anderenfalls wird eine leere Zeichenfolge zurückgegeben. *ContentType* ist eine schreibgeschützte Eigenschaft.

Die möglichen Werte von „ContentType“ in einer Standardinstallation sind:

Wert	Beschreibung
EMAIL_ADDRESS	Das Feld enthält eine E-Mail-Adresse.
PHONE	Das Feld enthält eine Telefonnummer.
CELLULAR	Das Feld enthält eine Mobiltelefonnummer.
URL	Das Feld enthält eine URL, z. B. die Adresse einer Firmen-Homepage.

string **Datatype**

Gibt den Datentyp des Feldes zurück. Ein Satz von **Datentypen** wird für die Verwendung mit Kundendomänen definiert. Schreibgeschützt.

string **Description**

Gibt die Beschreibung des Datenfeldes zurück, wie in der Kundendomäne definiert. Schreibgeschützt.

string **FieldName**

Gibt den Namen des Datenfeldes zurück, wie in der Kundendomäne definiert. Schreibgeschützt.

variant **LinkedCustDomainID**

Wenn das Feld eine Kunden-ID in einer anderen (verknüpften) Kundendomäne darstellt, ist *LinkedCustDomainID* die eindeutige ID dieser Kundendomäne. Anderenfalls ist *LinkedCustDomainID null*.

bool **Modified**

Zeigt an, ob der Feldwert geändert wurde. Schreibgeschützt.

variant **NewValue**

Gibt den neuen Wert eines geänderten Feldes zurück.

Hinweis: Diese Eigenschaft ist *nicht* schreibgeschützt und kann durch Kundendaten-Plug-Ins festgelegt werden, um Änderungen in der zugrunde liegenden Datenquelle widerzuspiegeln. Beispiel: Wenn ein Wert in einer ID-Spalte in einem Kunden-Plug-In generiert wird, kann das Plug-In *NewValue* festlegen, so dass er später von Dialogue Server oder anderen Plug-Ins in derselben Transaktion verwendet werden kann. Dies kann notwendig sein, weil Dialogue Server die Datengruppe aus der zugrunde liegenden Datenquelle nicht erneut abruft, nachdem die Methoden *InsertRow( )* oder *UpdateRow( )* des Kundendaten-Plug-Ins ausgeführt wurden.

variant **OldValue**

Gibt den alten Wert eines geänderten Feldes zurück. Schreibgeschützt.

bool **ReadOnly**

Zeigt an, ob das Feld schreibgeschützt ist. Schreibgeschützt.

integer **Size**

Gibt die Größe des Feldes zurück, wie in der Kundendomäne definiert. *Size* ist für Zeichenfolgefelder gültig. Schreibgeschützt.

string **SourceFieldName**

Gibt den Quellnamen des Datenfeldes zurück, wie in der Kundendomäne definiert. Schreibgeschützt.

Methoden

Es sind keine Methoden verfügbar.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDataFields-Schnittstelle

Beschreibung

IMHDataFields bietet Zugriff auf Felder und Feldwerte in einer Datengruppe einer Kundendomäne.

Anwendung

„IMHDataFields“ wird in Kundendaten-Plug-Ins verwendet, wenn Kundendaten aktualisiert werden.

Übergeordnete Schnittstelle

„IMHDataFields“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Die Anzahl der verfügbaren Felder. Schreibgeschützt.

IMHDataField **Field**[*integer* Index]

Bietet Zugriff auf ein Objekt, das ein einzelnes Datenfeld darstellt. Schreibgeschützt.

Index ist eine ganze Zahl, die verwendet wird, um auf das Feld zuzugreifen. Die erste Feld hat einen Indexwert von 0.

IMHDataField **FieldByName**[*string* FieldName]

Bietet Zugriff auf ein Objekt, das ein einzelnes Datenfeld darstellt. Schreibgeschützt.

FieldName ist der logische Name des Feldes, wie in der Kundendomäne definiert.

string **DataGroupName**

Der Name der Datengruppe in der Kundendomäne. Schreibgeschützt.

IMHDataField **FieldBySourceName**[*string* SourceFieldName]

Bietet Zugriff auf ein Objekt, das ein einzelnes Datenfeld darstellt. Schreibgeschützt.

SourceFieldName ist der Quellfeldname des Feldes, wie in der Kundendomäne definiert.

Methoden

Es sind keine Methoden verfügbar.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDataGroupPlugin-Schnittstelle

Beschreibung

IMHDataGroupPlugin enthält Methoden, die von Kundendaten-Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die Logik, die Kundendomänendaten aktualisiert, wird durch „IMHDataGroupPlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHDataGroupPlugin“ erbt von der Schnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **CheckForDuplicates**(**IMHCustomer** Customer, **IMHDuplicateList** DuplicateList)

CheckForDuplicates wird von Dialogue Server aufgerufen, wenn eine Prüfung auf Duplikate angefordert wird. Diese Methode implementiert die eigentliche Suche nach potentiellen Duplikaten in der Kundendomäne (oder der Kundendatenbank).

Customer ist ein Objekt, das den Kunden darstellt, für den die Prüfung auf Duplikate angefordert wird. Beispiel: Wenn ein neuer Kunde in Customer View eingefügt wird und eine Prüfung auf Duplikate angefordert wird, enthält das Objekt *Customer* Informationen über den einzufügenden Kunden.

DuplicateList ist ein Objekt, das eine Liste der gefundenen potentiellen Duplikate enthält. Wenn *CheckForDuplicates* implementiert wird, muss die Methode *DuplicateList.AddDuplicate(...)* aufgerufen werden, um die Kunden-ID jedes gefundenen potentiellen Duplikats hinzuzufügen.

void **DeleteRow**(**IMHCustomer** Customer, **IMHDataFields** Fields)

DeleteRow wird von Dialogue Server aufgerufen, wenn eine Zeile einer Datengruppe in einer Kundendomäne gelöscht werden soll. Die Zeile ist immer mit einem einzelnen Kunden verbunden. Wenn sich die Zeile in der Hauptdatengruppe der Domäne befindet, wird der Kunde gelöscht.

Customer ist ein Objekt, das den Kunden darstellt.

Fields ist ein Objekt, das Informationen über Felder der zu ändernden Datengruppe enthält.

variant **InsertRow**(**IMHCustomer** Customer, **IMHDataFields** Fields)

InsertRow wird von Dialogue Server aufgerufen, wenn eine Zeile einer Datengruppe in einer Kundendomäne hinzugefügt werden soll. Die Zeile ist immer mit einem einzelnen Kunden verbunden. Wenn sich die Zeile in der Hauptdatengruppe der Domäne befindet, wird ein neuer Kunde eingefügt. In diesem Fall muss der zurückgegebene Wert die ID des neuen Kunden sein. Ansonsten gibt „InsertRow“ *null* zurück.

Customer ist ein Objekt, das den Kunden darstellt.

Fields ist ein Objekt, das Informationen über Felder der zu ändernden Datengruppe enthält.

void **UpdateRow**(**IMHCustomer** Customer, **IMHDataFields** Fields)

UpdateRow wird von Dialogue Server aufgerufen, wenn eine Zeile einer Datengruppe in einer Kundendomäne aktualisiert werden soll. Die Zeile ist immer mit einem einzelnen Kunden verbunden.

Customer ist ein Objekt, das den Kunden darstellt.

Fields ist ein Objekt, das Informationen über Felder der zu ändernden Datengruppe enthält.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDialog-Schnittstelle

Beschreibung

IMHDialog bietet Zugriff auf Informationen zu einem Dialog.

Anwendung

IMHDialog wird in Verzweigungs-Plug-Ins verwendet.

Übergeordnete Schnittstelle

IMHDialog erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **AllowMulti**

Zeigt an, ob das mehrfache Auftreten von Kunden im Dialog erlaubt ist. Wenn *true* gesetzt ist, werden die Kontextwerte der Teilnehmer verwendet. Schreibgeschützt.

integer **CustDomainID**

Gibt die ID (eindeutige Kennung) der aktuellen Kundendomäne zurück. Schreibgeschützt.

string **DialogPathUNC**

Die UNC des Dialogpfades. Schreibgeschützt.

integer **GroupCount**

Gibt die Anzahl der Gruppen im Dialog zurück.

IMHDialogGroup **Groups**[*integer* GroupIndex]

Bietet Zugriff auf eine Gruppe von Dialogen.

GroupIndex ist der Index der Gruppe im Dialog. Die Indizierung ist null-basiert, beginnend mit „0“ für die erste Gruppe.

integer **ID**

Bietet Zugriff auf die eindeutige ID des Dialogs. Schreibgeschützt.

string **LogPathUNC**

Die UNC des Standardprotokollpfads des Dialogs. Schreibgeschützt.

string **Messages PathUNC**

Die UNC des Nachrichtenpfads des Dialogs. Nachrichten, die durch den Dialog produziert und als Dateien gespeichert wurden, werden standardmäßig in diesem Verzeichnis abgelegt. Schreibgeschützt.

string **Name**

Name ist der Name des Dialogs. Schreibgeschützt.

integer **OperationCount**

Gibt die Anzahl der Vorgänge im Dialog zurück.

IMHDialogOperation Operations[*integer* OperationIndex]

Bietet Zugriff auf eine Operation im Dialog.

OperationIndex ist der Index der Operation im Dialog. Die Indizierung ist null-basiert, beginnend mit „0“ für die erste Gruppe.

string ***Reporting*** **PathUNC**

Die UNC des Berichtspfads des Dialogs. Dies ist das Standardverzeichnis für die Speicherung von Berichtsdateien, die vom Dialog produziert werden. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHAnswerFormList GetAnonymousAnswerForms(integer QuestionnaireID, integer MaxCount)

Gibt eine Liste aller anonymen Antwortformulare zu einem bestimmten Fragebogen zurück, die im Dialog noch nicht als verarbeitet markiert wurden.

„QuestionnaireID“ ist die eindeutige ID des Fragebogens.

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

Hinweis: Ein Antwortformular kann in einem Dialog als verarbeitet markiert werden, indem die Methode *MarkAsProcessedInDialog( )* des Antwortformulars aufgerufen wird (siehe **IMHAnswerForm**).

IMHDialogGroup GetGroupByID(*integer* GroupID)

Bietet Zugriff auf eine Gruppe im Dialog, indem die eindeutige ID der Gruppe angegeben wird.

GroupID ist die eindeutige ID der Gruppe und entspricht der Datenbankspalte DLG_GROUP.DG_ID.

IMHAnswerFormList GetIdentifiedAnswerForms(integer QuestionnaireID, integer MaxCount)

Gibt eine Liste identifizierter Antwortformulare zu einem bestimmten Fragebogen zurück, die im Dialog noch nicht als verarbeitet markiert wurden. Es werden nur Antwortformulare zurückgegeben, die mit Kunden in der Kundendomäne des Dialogs verbunden sind.

„QuestionnaireID“ ist die eindeutige ID des Fragebogens.

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

Hinweis: Ein Antwortformular kann in einem Dialog als verarbeitet markiert werden, indem die Methode *MarkAsProcessedInDialog( )* des Antwortformulars aufgerufen wird (siehe **IMHAnswerForm**).

IMHAnswerFormList GetIdentifiedAnswerFormsEx(integer QuestionnaireID, bool IncludeComplete, bool IncludeUncomplete, integer MaxCount)

Gibt eine Liste identifizierter Antwortformulare zu einem bestimmten Fragebogen zurück, die im Dialog noch nicht als verarbeitet markiert wurden. Es werden nur Antwortformulare zurückgegeben, die mit Kunden in der Kundendomäne des Dialogs verbunden sind.

„QuestionnaireID“ ist die eindeutige ID des Fragebogens.

IncludeComplete legt fest, dass ausschließlich als abgeschlossen markierte Antwortformulare zurückgegeben werden.

IncludeIncomplete legt fest, dass ausschließlich nicht als abgeschlossen markierte Antwortformulare zurückgegeben werden.

Hinweis: Sie können nicht gleichzeitig *IncludeComplete* und *IncludeIncomplete* auf **false** setzen.

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

Hinweis: Ein Antwortformular kann in einem Dialog als verarbeitet markiert werden, indem die Methode *MarkAsProcessedInDialog( )* des Antwortformulars aufgerufen wird (siehe **IMHAnswerForm**).

IMHDialogOperation GetOperationByID(*integer* OperationID)

Bietet Zugriff auf eine Operation im Dialog, indem die eindeutige ID der Operation angegeben wird.

OperationID ist die eindeutige ID der Operation und entspricht der Datenbankspalte DLG_OPERATION.DO_ID.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDialogUtils-Schnittstelle

Beschreibung

IMHDialogUtils bietet Zugriff auf Dialoge.

Anwendung

„IMHDialogUtils“ wird in Plug-Ins zum Arbeiten mit Dialogen verwendet.

Übergeordnete Schnittstelle

„IMHDialogUtils“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHDialog GetDialog(*integer* DialogID)

Gibt ein Objekt zurück, das einen bestimmten Dialog repräsentiert.

DialogID ist die eindeutige ID des zurückzugebenden Dialogs.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDialogGroup-Schnittstelle

Beschreibung

IMHDialogGroup ermöglicht den Zugriff auf Informationen über eine Operation in einem Dialog.

Anwendung

„IMHDialogGroup“ wird in Verzweigungs-Plug-Ins verwendet.

Übergeordnete Schnittstelle

„IMHDialogGroup“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

IMHDialog Dialog

Bietet Zugriff auf ein Objekt, das den Dialog repräsentiert, zu dem die Gruppe gehört.

string **GroupDataType**

Ermöglicht den Zugriff auf den grundlegenden Typen einer Dialoggruppe. *GroupDataType* kann die folgenden Zeichenfolgenwerte haben:

Wert	Beschreibung
Datenbank	Der Database-Datentyp steht für alle Kunden in Ihrer Kundendatenbank, also für alle Kunden, die

	in der Kundendomäne durch den Dialog angesprochen werden.
Standard	Der Standard-Datentyp ist der für Dialogteilnehmer normalerweise verwendete Datentyp.
Fuzzy	Der Fuzzy-Datentyp wird in Gruppentypen verwendet, die für nicht identifizierte Kunden stehen, z. B. ein Marktplatz oder ein Zielgruppenpublikum.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Operation. Schreibgeschützt.

string **Name**

Name ist der Name der Operation. Schreibgeschützt.

IMHDialogOperation OutOperation

Ermöglicht den Zugriff auf die Operation zum Verschieben von Teilnehmern aus der Gruppe. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

bool **InsertParticipant**(**IMHCustomer** Customer)

Fügt einen Kunden als Teilnehmer in die Dialoggruppe ein. Gibt den Wert *true* (wahr) zurück, wenn der neue Teilnehmer erfolgreich eingefügt wurde. Wenn der Kunde (einschließlich Kontext) bereits Mitglied des Dialogs ist, wird *false* zurückgegeben.

Customer ist ein Objekt, das für den Kunden steht.

bool **InsertParticipantByID**(*string* CustomerID, *string* Context)

Entspricht *InsertParticipant( )*, außer dass IDs anstelle eines für den Kunden stehenden Objektes als Parameter verwendet werden.

CustomerID ist die eindeutige ID des Kunden, der als Dialogteilnehmer eingetragen werden soll.

Context ist der Zeichenfolgenkontextwert, der bei einigen Dialogen verwendet wird. Falls der Kontext nicht verwendet wird, setzen Sie diesen Parameter auf den Zeichenfolgenwert „0“.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDialogOperation-Schnittstelle

Beschreibung

IMHDialogOperation ermöglicht den Zugriff auf Informationen über eine Operation in einem Dialog.

Anwendung

„IMHDialogOperation“ wird in Verzweigungs-Plug-Ins verwendet.

Übergeordnete Schnittstelle

„IMHDialogOperation“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **BranchCount**

Gibt die Anzahl der Verzweigungen in der Operation zurück.

IMHBranchInfo **Branches**[*integer* BranchIndex]

Ermöglicht den Zugriff auf eine Verzweigung in einer Operation im Dialog.

BranchIndex ist der Index der Verzweigung in der Operation. Die Indexierung hat die Basis null („0“) und beginnt entsprechend mit „0“ für die Verzweigung mit Priorität 1.

variant **CustomValue**[*variant* ValueName]

Ermöglicht den Zugriff auf eine Name/Wert-Sammlung benutzerdefinierter Werte. Ein benutzerdefinierter Wert kann aus verschiedenen einfachen Datentypen bestehen, z. B. Zeichenfolge (String), Ganzzahl (Integer), boolescher Wert (Boolean) oder Gleitkommazahl (Float).

Mithilfe von benutzerdefinierten Werten können Sie Werte in einer Verzweigung einer Operation festlegen und dann auf diese in einer anderen Verzweigung zugreifen. Benutzerdefinierte Werte befinden sich während der Ausführung einer Operation im Arbeitsspeicher und werden nicht in der Datenbank gespeichert.

IMHDialogGroup **FromGroup**

Ermöglicht den Zugriff auf die Absendergruppe der Operation, d. h. die Gruppe, von welcher aus die Teilnehmer durch die Operation verschoben werden.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Operation. Schreibgeschützt.

string **Name**

Name ist der Name der Operation. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

integer **Execute**(*integer* MaxCount)

Dadurch wird die Operation sofort ausgeführt und auf den Abschluss der Operation gewartet. Es wird die Gesamtanzahl der durch die Operation verschobenen Teilnehmer zurückgegeben.

MaxCount stellt die maximale Anzahl an Teilnehmern dar, für welche die Operation weiter ausgeführt wird. Legen Sie *MaxCount* auf den Wert „-1“ fest, damit die Operation für alle Teilnehmer der Absendergruppe ausgeführt wird.

Hinweis: Die Operation wird in derselben, momentan laufenden Transaktion ausgeführt, dadurch wird z. B. zum Ausführen der Operation keine neue Transaktion gestartet.

void **ExecuteAsync**(*integer* MaxCount)

Fordert Dialogue Server zur Ausführung der Operation auf und kehrt zurück, ohne auf den Start der Ausführung der Operation zu warten.

MaxCount stellt die maximale Anzahl an Teilnehmern dar, für welche die Operation weiter ausgeführt wird. Legen Sie *MaxCount* auf den Wert „-1“ fest, damit die Operation für alle Teilnehmer der Absendergruppe ausgeführt wird.

Hinweis: Die Operation wird in einer separaten Transaktion ausgeführt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDialogServerServices-Schnittstelle

Beschreibung

Die Schnittstelle *IMHDialogServerServices* unterstützt allgemeine Funktionen in Dialogue Server.

Anwendung

„IMHDialogServerServices“ wird durch das Objekt *DialogServer* unterstützt, das in allen Plug-Ins als Toolbox für Plug-In-Implementierer verfügbar ist.

Übergeordnete Schnittstelle

„IMHDialogServerServices“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ApplicationSystem**

Bietet Zugriff auf den Namen des Anwendungssystems der aktuellen Benutzersitzung. Das Anwendungssystem wird angegeben, wenn sich ein Benutzer bei Dialogue Server anmeldet. Schreibgeschützt.

IMHContentObjectUtils ContentObjects

Bietet Zugriff auf Methoden, die mit Inhaltsobjekten verbunden sind.

IMHDialogUtils Dialogs

Bietet Zugriff auf Methoden für den Zugriff auf Dialoge.

string **InstanceName**

Gibt den Namen der aktuellen Instanz von Dialogue Database an. Schreibgeschützt.

string **LanguageCode**

Gibt den Sprachcode der aktuellen Benutzersitzung an, z. B. „en-US“ (amerikanisches Englisch) oder „de-DE“ (Französisch). Schreibgeschützt.

IMHLicenseInfo LicenseInfo

Bietet Zugriff auf Informationen zur aktuellen Lizenz.

IMHMessageUtils Messages

Bietet Zugriff auf Methoden für den Zugriff auf Nachrichten und Nachrichtenvorlagen.

IMHQuestionnaireUtils Questionnaires

Bietet Zugriff auf Methoden für den Zugriff auf Fragebögen und Antwortformulare.

IMHReportEngine ReportEngine

Bietet Zugriff auf ein Objekt, welches das Berichtsmodul innerhalb von Dialogue Server repräsentiert. Schreibgeschützt.

ReportEngine besitzt Methoden, um Berichte zu erstellen und darauf zuzugreifen.

IMHSystemUser User

Bietet Zugriff auf ein Objekt, das den aktuellen Benutzer repräsentiert. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHWebPublicFile AddWebPublicFile(*string* SourceUNC, *string* Description)

Veröffentlicht eine Datei in Dialogue Server, so dass auf diese durch eine URL zur Anwendung „Web Utilities“ zugegriffen werden kann. Dateien, die für das Internet veröffentlicht wurden, werden in der Datenbanktabelle *WEB_PUBLIC_FILE* gespeichert.

AddWebPublicFile(...) gibt ein Objekt zurück, das die Datei repräsentiert.

SourceUNC ist der Pfad und der Dateiname der lokalen zu veröffentlichenden Datei.

Description ist eine optionale Dateibeschreibung, die in Dialogue Database mit der Datei gespeichert wird.

string **BuildTrackedLinkUrl**(*string* OriginalUrl, *string* LinkName, *bool* NoTrack, *bool* TrackAnonymously, *variant* CustomerMessageID)

Verpackt eine URL in eine neue, welche die Verknüpfungsnachverfolgung durch Verwendung der Anwendung „Web Utilities“ unterstützt. Die URL zur Nachverfolgung wird zurückgegeben.

OriginalUrl ist die nachzuverfolgende URL.

LinkName ist der Name, welcher der URL zu Berichtszwecken gegeben wird.

NoTrack deaktiviert die Verknüpfungsnachverfolgung. Diese Option wird von Dialogue Server intern verwendet.

Wenn *TrackAnonymously* auf *true* gesetzt ist, wird jede Nachverfolgung in der Tabelle WEB_TRACK_LOG gespeichert. Die protokollierten Informationen sind jedoch nicht mit einer bestimmten Nachricht oder einem bestimmten Kunden verbunden. Ist der Wert auf *false* gesetzt, muss eine Kundennachricht-ID angegeben werden, um eine bestimmte Nachricht oder einen bestimmten Kunden nachzuverfolgen.

CustomerMessageID ist die ID der Nachricht, über die mit einem Kunden kommuniziert wird. Sie entspricht dem Wert in der Spalte CM_ID der Tabelle CUSTOMER_MESSAGE.

variant **ExecutePlugin**(*string* PluginName, *array* Params)

Führt ein Dialogue Server-Plug-In vom Typ *Generisches Plug-In* aus, das die Schnittstelle **IMHGenericPlugin** implementiert. *PluginName* ist der in Dialogue Admin *definierte Name des Plug-Ins. Params* ist ein Array, das Parameterwerte enthält.

Unter dem Rückgabewert versteht man den vom Plug-In zurückgegebenen Wert.

IMHCustomerContainer **GetCustomers**(*integer* CustDomainID, *string* FilterExpression, *string* ContextExpression, *string* DataFields, *integer* MaxCount)

Gibt ein Objekt zurück, das die Menge der Kunden repräsentiert, welche die angegebenen Kriterien erfüllt.

CustDomainID ist die ID der Kundendomäne.

Der Ausdruck *FilterExpression* wird verwendet, um die Auswahl zurückgegebener Kunden zu filtern oder zu verfeinern.

ContextExpression weist, wenn angegeben, dem Feld *mh_context* einen Wert für jeden Kunden zu. „ContextExpression“ ist ein Ausdruck, der eine Zeichenfolge oder ein Array von Zeichenfolgen zurückgibt. Im Fall eines Arrays von Zeichenfolgen können mehrere Kontextwerte pro Kunde existieren und der Kunde kann mehrere Male (nämlich einmal pro Kontextwert) gezählt werden.

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen Objekt zwischengespeichert wird. Auf andere Felder kann ebenfalls zugegriffen werden. Das Angeben von „DataFields“ optimiert jedoch die Datenübertragung in Dialogue Server.

MaxCount zeigt die maximale Anzahl zurückzugebender Kunden an. MaxCount muss auf -1 eingestellt werden, um alle Kunden zurückzugeben, die die anderen angegebenen Kriterien erfüllen.

IMHCustomerContainer GetCustomersEx(*integer* CustDomainID, *string* SQLStatement, *array* Param, *bool* ContextIncluded, *string* ConnectionName, *string* DataFields, *integer* MaxCount)

Gibt ein Objekt zurück, das eine Menge von Kunden repräsentiert, die den in einer SQL-Anweisung zurückgegebenen IDs entsprechen.

CustDomainID ist die ID der Kundendomäne.

SQLStatement ist eine SQL-SELECT-Anweisung. Die SELECT-Anweisung muss eine Spalte namens *mh_customer_id zurückgeben, welche die Kunden-IDs enthält. Optional kann eine Spalte namens mh_context* zurückgegeben werden, die Kontextwerte enthält.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ContextIncluded gibt an, ob die SELECT-Anweisung die Spalte *mh_context* zurückgibt.

ConnectionName ist ein Name, der sich auf eine *sekundäre Datenbank* bezieht, die in Dialogue Admin definiert ist. Setzen Sie *„ConnectionName“* auf *null*, um die Standardverbindung zu verwenden.

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen Objekt zwischengespeichert wird. Auf andere Felder kann ebenfalls zugegriffen werden. Das Angeben von „DataFields“ optimiert jedoch die Datenübertragung in Dialogue Server.

MaxCount zeigt die maximale Anzahl zurückzugebender Kunden an. MaxCount muss auf -1 eingestellt werden, um alle Kunden zurückzugeben, die die anderen angegebenen Kriterien erfüllen.

IMHCustomerContainer GetCustomerSelection(*integer* SelectionID, *string* DataFields, *integer* MaxCount)

Gibt ein Objekt zurück, das eine Menge von Kunden in der angegebenen Auswahl repräsentiert.

SelectionID entspricht der ID der Auswahl.

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen Objekt zwischengespeichert wird. Auf andere Felder kann ebenfalls zugegriffen werden. Das Angeben von „DataFields“ optimiert jedoch die Datenübertragung in Dialogue Server.

MaxCount zeigt die maximale Anzahl zurückzugebender Kunden an. MaxCount muss auf -1 eingestellt werden, um alle Kunden zurückzugeben, die die anderen angegebenen Kriterien erfüllen.

IMHCustomerContainer GetCustomerSelectionSorted(*integer* SelectionID, *string* DataFields, *string* SortFields, *integer* MaxCount)

Wie *GetCustomerSelection(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

IMHCustomerContainer GetCustomersSorted(*integer* CustDomainID, *string* FilterExpression, *string* ContextExpression, *string* DataFields, string SortFields *, integer* MaxCount)

Wie *GetCustomers(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

IMHCustomerContainer GetCustomersSortedEx(*integer* CustDomainID, *string* SQLStatement, *array* Param, *bool* ContextIncluded, *string* ConnectionName, *string* DataFields, *string* SortFields, *integer* MaxCount)

Bewirkt dasselbe wie *GetCustomersEx(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

string **GetDataFieldsInExpression(** *integer* CustDomainID, *string* Expression **)**

Gibt eine durch Semikola getrennte Liste der Domänenfelder zurück, die zum Auswerten eines Ausdrucks benötigt werden.

CustDomainID ist die eindeutige ID der Kundendomäne.

Expression ist der zu analysierende Ausdruck.

int64 **GetSequenceID**(*string* SequenceName)

Gibt den nächsten Wert der angegebenen Sequenz zurück. *Sequences* ist ein von einem Dialogue Server zur Verfügung gestellter Mechanismus zur Aufrechterhaltung eindeutiger Kennungen. Durch den Dialogue Server wird die Einzigartigkeit der zurückgegebenen Zahl in unterschiedlichen Aufrufen der „GetSequenceID“ garantiert.

SequenceName ist der Name der Sequenz. Wenn eine Sequenz festgelegt wird, die nicht existiert, wird automatisch eine neue Sequenz erstellt.

Hinweis: Sequenzen sind interne Mechanismen in Dialogue Server und sollten nicht mit Datenbanksequenzen verwechselt werden. Zum Beispiel verfügt Oracle über einen eigenen, als *Sequenzen* bezeichneten Mechanismus.

IMHCustomer GetSingleCustomer(*integer* CustDomainID, *string* CustomerID, *string* Context, *string* DataFields)

Gibt einen Verweis zu einem Objekt zurück, das einen einzelnen Kunden in der angegebenen Kundendomäne repräsentiert.

CustDomainID ist die eindeutige ID der Kundendomäne. *CustomerID* ist die eindeutige ID eines Kunden in dieser Domäne.

Context ist der Kontextwert des Kunden. Wenn kein Kontext gefordert wird, Context auf die Zeichenfolge "0" einstellen.

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen Objekt zwischengespeichert wird. Auf andere Felder kann ebenfalls zugegriffen werden. Das Angeben von „DataFields“ optimiert jedoch die Datenübertragung in Dialogue Server.

IMHSQLDef GetSQLDef(*string* SQLName)

Bietet Zugriff auf ein Objekt, das im SQL-Repository von Dialogue Admin eine SQL-Anweisung repräsentiert. *SQLName* ist der technische Name der Anweisung.

IMHWebPublicFile GetWebPublicFile(*string* ContentID)

Gibt ein Objekt zurück, das eine im Internet veröffentlichte Datei repräsentiert.

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

void **LogDebugMessage**(*string* Message)

Protokolliert eine Nachrichtenzeichenfolge in Dialogue Manager Service, die von Process Monitor überwacht werden kann.

IMHCustomer NewCustomer(*integer* CustDomainID, *string* DataGroups)

Startet das Einfügen eines neuen Kunden in eine Kundendomäne.

CustDomainID ist die eindeutige ID der Kundendomäne, welcher der neue Kunde angehören wird.

DataGroups ist eine durch Semikola getrennte Liste von Kundendatengruppen (aus der Kundendomänendefinition), die bearbeitbar ist, wenn einem neuen Kunden Daten zugewiesen werden.

Nach dem Aufrufen von *NewCustomer()* werden die Methoden *IMHCustomer.SetFieldValue()* und *IMHCustomer.PostChanges()* zum Speichern der Daten des neuen Kunden gespeichert. Ein Beispiel finden Sie unter **IMHCustomer**.

bool **ParamAsBool**(*string* CollectionName, *string* ParamName)

Liefert einen booleschen Parameterwert aus Dialogue Server, wobei die Parametersammlung und der Parametername als Eingabe verwendet werden. Parameter werden in Dialogue Admin eingerichtet.

float **ParamAsFloat**(*string* CollectionName, *string* ParamName)

Liefert eine Gleitkommazahl als Parameterwert aus Dialogue Server, wobei die Parametersammlung und der Parametername als Eingabe verwendet werden. Parameter werden in Dialogue Admin eingerichtet.

integer **ParamAsInteger**(*string* CollectionName, *string* ParamName)

Liefert einen ganzzahligen Parameterwert aus Dialogue Server, wobei die Parametersammlung und der Parametername als Eingabe verwendet werden. Parameter werden in Dialogue Admin eingerichtet.

string **ParamAsString**(*string* CollectionName, *string* ParamName)

Liefert eine Zeichenfolge als Parameterwert aus Dialogue Server, wobei die Parametersammlung und der Parametername als Eingabe verwendet werden. Parameter werden in Dialogue Admin eingerichtet.

string **ScrambleURL**(*string* Url)

Verschlüsselt eine URL. *Url* ist die zu verschlüsselnde URL.

void **SendAdminEmail**(*string* ToAddress, *string* FromAddress, *string* Subject, *string* MessageBody, *string* Attachments)

Versendet eine E-Mail unter Verwendung des in Dialogue Admin konfigurierten E-Mail-Ausgangskanals. Wenn das Senden der E-Mail fehlschlägt, wird eine Ausnahme ausgelöst.

ToAddress ist der Empfänger der E-Mail.

FromAddress ist der Absender der E-Mail.

Subject ist der E-Mail-Betreff.

MessageBody ist der Textinhalt der E-Mail.

Attachments ist eine durch Semikola getrennte Liste der an die E-Mail angehängten Dateien.

In den obigen Adressparametern können mehrere Adressen durch Semikola getrennt werden.

Hinweis: Diese Methode ist für administrative Nachrichten konzipiert, nicht für die Massenkommunikation mit dem Endkunden.

void **SendAdminSMS**(*string* ToNumber, *string* FromNumber, *string* MessageText)

Versendet eine SMS-Nachricht unter Verwendung des in Dialogue Admin konfigurierten E-Mail-Ausgangskanals. Wenn das Senden der Nachricht fehlschlägt, wird eine Ausnahme ausgelöst.

ToNumber ist die Mobiltelefonnummer des Empfängers.

FromNumber ist die Nummer oder der Text, der den Absender der SMS-Nachricht identifiziert.

MessageText ist der Textinhalt der SMS-Nachricht.

Hinweis: Diese Methode ist für administrative Nachrichten konzipiert, nicht für die Massenkommunikation mit dem Endkunden.

datetime **ServerDateTime**( )

Liefert den (aktuellen) Datums-/Zeitwert des Systems aus Dialogue Server.

string **ServerSession**( )

Gibt die aktuelle Serversitzung zurück.

Die Zeichenfolge „ServerSession“ kann zum Aufrufen der Dialogue Server-API verwendet werden.

void **Sleep**(*integer* Milliseconds)

Versetzt dem aktuellen Ausführungsthread für einen angegebenen Zeitraum in den Ruhemodus. Zum Implementieren dieser Methode wird die standardmäßige Windows-API verwendet.

Milliseconds ist die Dauer des Ruhemodus in Millisekunden (ms).

object **SQLConnectionObject**(*string* ConnectionName)

Gibt eine Referenz zu einem ADO-Verbindungsobjekt zurück. *ConnectionName* ist der Name der in Dialogue Admin definierten sekundären Datenbank. Wenn Sie dieses Arguments auf *null* setzen, wird das Standardverbindungsobjekt zurückgegeben, das verwendet wird, um auf die Datenbank von Dialogue Server zuzugreifen.

integer **SQLExecute**(*string* SQLName, *array* Params)

Führt eine im Dialogue Admin SQL Repository gespeicherte SQL-Anweisung aus. *SQLName* ist der technische Name der Anweisung. *Params* ist ein Array, das Parameterwerte enthält.

„SQLExecute“ gibt die Anzahl der betroffenen Datensätze zurück.

integer **SQLExecuteRaw**(*string* SQLStatement, *array* Params, *string* ConnectionName)

Führt die in „SQLStatement“ angegebene SQL-Anweisung aus. *Params* ist ein Array, das Parameterwerte enthält. *ConnectionName* ist ein Name, der sich auf eine sekundäre Datenbank bezieht, die in Dialogue Admin definiert ist. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

„SQLExecute“ gibt die Anzahl der betroffenen Datensätze zurück.

void **SQLExecuteStoredProc**(*string* StoredProcName, *array* Params, *string* ConnectionName)

Führt eine gespeicherte Prozedur mit der angegebenen Datenbankverbindung durch.

„StoredProcName“ ist der in der Datenbank definierte Name der gespeicherten Prozedur, während *Params* ein Array mit Parameterwerten ist.

ConnectionName ist der technische Name der in Dialogue Admin definierten sekundären Datenbankverbindung. Das Leerlassen von „ConnectionName“ zeigt an, dass die Standardverbindung der Datenbank verwendet werden soll.

string **SQLGetDatasetXML**(*string* SQLName, *array* Params, *integer* MaxRows)

Gibt einen in Dialogue Admin als XML definierten Datensatz zurück.

SQLName ist der Name des Elements im SQL-Repository, das den Datensatz definiert.

Params ist eine Matrix, die Parameterwerte enthält (Host-Variablen der Datenbank).

MaxRows gibt die maximale Anzahl abzurufender Zeilen an. Wenn MaxRows auf -1 festgelegt wird, werden alle Zeilen zurückgegeben, während bei einem entsprechenden Wert von 0 nur das XML-Schema zurückgegeben wird.

int64 **SQLGetLastAutoIncID**(*string* TableName)

Gibt den letzten automatisch inkrementierten Wert zurück, wenn Zeilen in Dialogue Database eingefügt werden. Die Methode wird nur bei Tabellen unterstützt, die Masseneinfügungen unterstützen.

TableName ist der Name der Datenbanktabelle, in welche die Zeile eingefügt wurde.

Hinweis: *SQLGetLastAutoIncID( )* arbeitet anders bezüglich der mit Dialogue Server verwendeten Datenbankplattform.

Server

Auf SQL Server ist das Aufrufen von *SQLGetLastAutoIncID( )* gleichwertig zur SELECT-Anweisung „select@@identity“. Der Parameter *TableName* wird auf SQL Server nicht verwendet.

ORACLE DBMS

Auf Oracle DBMS werden Sequenzen verwendet, um in Dialogue Server eindeutige IDs zu generieren. Die verwendeten Sequenzen tragen folgende Namen: <Tabellenname>_SEQ (z. B. DLG_PARTICIPANT_SEQ).

object **SQLOpen**(*string* SQLName, *array* Params)

Öffnet einen ADO-Datensatz, und gibt diesen unter Verwendung der im SQL-Repository von Dialogue Admin gespeicherten SQL-Anweisung zurück. *SQLName* ist der technische Name der Anweisung. *Params* ist ein Array, das Parameterwerte enthält.

object **SQLOpenRaw**(*string* SQLStatement, *array* Params, *string* ConnectionName)

Öffnet einen ADO-Datensatz und gibt diesen unter Verwendung der in „SQLStatement“ angegebenen SQL-Anweisung zurück. *Params* ist ein Array, das Parameterwerte enthält. *ConnectionName* ist ein Name, der sich auf eine *sekundäre Datenbank* bezieht, die in Dialogue Admin definiert ist. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

variant **SQLRetrieveValue**(*string* SQLName, *array* Params)

Öffnet einen Datensatz und gibt den Wert der ersten Zeile in der ersten Spalte zurück. Verwendet eine SQL-Anweisung, die im SQL-Repository von Dialogue Admin gespeichert ist. *SQLName* ist der technische Name der Anweisung. *Params* ist ein Array, das Parameterwerte enthält.

variant **SQLRetrieveValueRaw**(*string* SQLStatement, *array* Params, *string* ConnectionName)

Öffnet einen Datensatz und gibt den Wert der ersten Zeile in der ersten Spalte zurück. Verwendet die in „SQLStatement“ verwendete SQL-Anweisung. *Params* ist ein Array, das Parameterwerte enthält. *ConnectionName* ist ein Name, der sich auf eine sekundäre Datenbank bezieht, die in Dialogue Admin definiert ist. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

variant **SQLUpdateDatasetXML**(*string* ServerSession, *string* SQLName, *array* Params, *string* XMLDiffGram)

Aktualisiert einen Datensatz, der vorher mit der *GetDataset*-Methode abgerufen wurde.

UpdateDataset gibt die letzte von der Datenbank generierte ID für die im Datensatz enthaltene Tabelle auf höchster Ebene zurück. Typischerweise handelt es sich dabei um den Wert einer automatisch inkrementierten Spalte oder einer Datenbanksequenz. Wenn die entsprechenden Spalten nicht in Dialogue Admin definiert wurden, ist der Rückgabewert *null.*

SQLName ist der Name des Elements im SQL-Repository, das den Datensatz definiert.

Params ist ein Array von Parameterwerten. Diese Parameterwerte müssen den Parameterwerten entsprechen, die festgelegt wurden, als der Originaldatensatz mit *GetDataset* abgerufen wurde.

XMLDiffGram ist ein XML-Dokument, das neue, aktualisierte oder gelöschte Datensätze enthält. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams.

string **Translate**(*string* Msg)

Übersetzt (lokalisiert) eine Zeichenfolge. Identisch mit *TranslateAndFormat(...)* weiter unten, unterstützt aber keine Formatierung.

string **TranslateAndFormat**(*string* Msg, *array* Params)

Übersetzt (lokalisiert) und formatiert eine Zeichenfolge.

Msg ist die zu übersetzende oder zu formatierende Eingabezeichenfolge.

Params ist ein Array von Werten, das während des Formatierens in die Zeichenfolge (*Msg*) eingefügt wird.

Beispiel:

Meldung: „[MSG01] Die Werte sind {0} und {1}“, Params: [123, 987]

„[MSG01]“ ist die für die Übersetzung (Lokalisierung) verwendete Kennung. „{0}“ und „{1}“ werden durch Werte aus dem Array *Params* ersetzt (Formatierung).

Hinweis: Der Übersetzungsmechanismus ist ein interner Mechanismus, der für die Lokalisierung der Standard-Skript-Plug-Ins verwendet wird und nicht für benutzerdefinierte oder Add-On-Plug-Ins anwendbar ist. Die Formatierungsfunktion kann allgemein verwendet werden.

Beispiele

Das folgende Beispiel zeigt, wie man eine Debug-Nachricht mit Process Monitor aufzeichnet:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   DialogServer.LogDebugMessage("Trying to connect to SMS service provider");

   ..............

}
```

Das nächste Beispiel zeigt, wie man eine SQL UPDATE-Anweisung ausführt:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   DialogServer.SQLExecuteRaw("UPDATE balder.person SET e_mail = :param1"
WHERE person_id = :param2",
      new Array("testperson@test.com", 100099),
      NULL);
   ..............
}
```

IMHDuplicateList-Schnittstelle

Beschreibung

IMHDuplicateList bietet Zugriff auf eine Liste der Kunden-IDs, die potentielle Duplikate darstellen.

Anwendung

„IMHDuplicateList“ wird in Kundendaten-Plug-Ins verwendet, wenn eine Überprüfung auf Duplikate angefordert wird.

Übergeordnete Schnittstelle

„IMHDuplicateList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Die Anzahl der verfügbaren Felder. Schreibgeschützt.

string **Duplicate**[*integer Duplicate*Index]

Bietet Zugriff auf die Kunden-ID eines angegebenen Duplikats. Schreibgeschützt.

DuplicateIndex ist eine ganze Zahl, die verwendet wird, um auf das Duplikat zuzugreifen. Das Duplikat hat einen Indexwert von 0.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddDuplicate**(*string* CustomerID

Fügt ein Duplikat zur Liste hinzu.

CustomerID ist die eindeutige ID des Kunden, der als potentielles Duplikat identifiziert wurde.

void **Clear( )**

Löscht die Liste.

Beispiele

Es sind keine Beispiele verfügbar.

IMHDynamicValuesList-Schnittstelle

Beschreibung

IMHDynamicValuesList ermöglicht den Zugriff auf eine Werteliste.

Anwendung

„IMHDynamicValuesList“ wird in Plug-Ins zum Zugriff auf eine Werteliste verwendet, die als ein Verzweigungsparameter vom Typ „Dynamische Liste Mehrfachauswahl“ gespeichert wurde.

Übergeordnete Schnittstelle

„IMHDynamicValuesList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ListAsString** Ermöglicht den Zugriff auf eine Liste von durch Komma getrennten Werten. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHEmailBounceCodeList-Schnittstelle

Beschreibung

IMHEmailBounceCodeList ermöglicht den Zugriff auf eine Liste von E-Mail-Unzustellbarkeitscodes.

Anwendung

„IMHEmailBounceCodeList“ wird in Plug-Ins zum Zugriff auf die E-Mail-Unzustellbarkeitscodes von unzustellbaren E-Mails verwendet. Diese Schnittstelle wird vom standardmäßigen „Select/Divide“ von Verzweigungs-Plug-Ins für unzustellbare E-Mails verwendet.

Übergeordnete Schnittstelle

„IMHEmailBounceCodeList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ListAsString** Ermöglicht den Zugriff auf eine Liste von durch Komma getrennten Codes unzustellbarer E-Mails. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHExprFunctionDef-Schnittstelle

Beschreibung

IMHExprFunctionDef bietet Zugriff auf die Definition einer benutzerdefinierten Ausdrucksfunktion.

Anwendung

„IMHExprFunctionDef“ wird verwendet, wenn eine benutzerdefinierte Ausdrucksfunktion implementiert wird.

Übergeordnete Schnittstelle

IMHExprFunctionDef erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Description**

Beschreibung der Ausdrucksfunktion. Dieser Text wird als Tipp in der Toolbox von Visual Dialogue angezeigt.

string **Name**

Der Name der Ausdrucksfunktion.

Hinweis: Wenn Sie eine Ausdrucksfunktion bei Verwendung eines Plug-Ins implementieren, wird der Name standardmäßig gleich dem Namen des Plug-Ins festgelegt.

string **ResultType**

Der Datentyp des von der Ausdrucksfunktion zurückgegebenen Wertes. Siehe **Ausdrucksdatentypen**.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddParam**(*string* ParamName, *string* DataTypes)

Fügt einen Eingabeparameter zur Definition einer Ausdrucksfunktion hinzu.

ParamName ist der Name des Parameters.

DataTypes enthält die erlaubten Datentypen des Parameters. Siehe **Ausdrucksdatentypen**. Wenn mehrere Datentypen erlaubt sind, werden diese durch Semikola getrennt.

Beispiele

Siehe **Ausdrucks-Plug-Ins**.

IMHExprFunctionPlugin-Schnittstelle

Beschreibung

IMHExprFunctionPlugin enthält Methoden, die durch Ausdrucks-Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Benutzerdefinierte Ausdrucksfunktionen werden durch „IMHExprFunctionPlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHExprFunctionPlugin“ erbt von der Basisschnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **DefineFunction**(**IMHExprFunctionDef** **FunctionDef**)

Diese Methode wird durch Dialogue Server aufgerufen, um die Definition der Ausdrucksfunktion abzurufen.

FunctionDef ist das Objekt, das die Definition der Funktion enthält.

variant **EvaluateFunction**(**IMHCustomer** Customer, *safearray of variant* Params)

Die Ausführungsmethode wird aufgerufen, wenn das Plug-In ausgeführt wird.

Customer ist ein Objekt, das den Kunden repräsentiert, für den die Funktion ausgeführt wird.

„Params“ ist ein Array von Parameterwerten. Das Array ist vom COM-Datentyp *safearray* (siehe Microsoft COM-Dokumentation).

Die Methode *EvaluateFunction* gibt einen Wert mit dem Datentyp zurück, der in der Eigenschaft **FunctionDef.ResultType** konfiguriert wurde.

Hinweis: Bei der Implementierung als Skript wird der Parameteraustausch vereinfacht. Das Skript deklariert Parameter als eigenständige Eingabeparameter der Methode *EvaluateFunction*.

Beispiele

Siehe **Ausdrucks-Plug-Ins**.

IMHGenericPlugin-Schnittstelle

Beschreibung

IMHGenericPlugin enthält Methoden, die von generischen Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die in einem generischen Plug-In enthaltene Logik wird durch „IMHGenericPlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHGenericPlugin“ erbt von der Schnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

variant **Execute**(*safearray of variant* Params)

Die Ausführungsmethode wird aufgerufen, wenn das Plug-In ausgeführt wird.

„Params“ ist ein Array von Parameterwerten. Das Array ist vom COM-Datentyp *safearray* (siehe Microsoft COM-Dokumentation).

Die Methode *Execute* kann jeden Wertetyp zurückgeben.

Hinweis: Bei der Implementierung als Skript wird der Parameteraustausch vereinfacht. Das Skript deklariert Parameter als eigenständige Eingabeparameter der Methode *Execute*. Siehe **Beispiel**.

Beispiele

Es sind keine Beispiele verfügbar.

IMHLicenseInfo-Schnittstelle

Beschreibung

IMHLicenseInfo bietet Zugriff auf Informationen über die Lizenz von Dialogue Server.

Anwendung

„IMHLicenseInfo“ wird in Plug-Ins verwendet, um auf Eigenschaften der aktuellen Lizenz zuzugreifen, beispielsweise der Name des Lizenzinhabers.

Übergeordnete Schnittstelle

„IMHLicenseInfo“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

bool **AdvancedEmailTesting**

Zeigt an, ob die Lizenz das erweiterte Testen von E-Mail-Vorlagen unterstützt. Dies ist ein gebührenpflichtiger Dienst. Schreibgeschützt.

bool **DemoMode**

Zeigt an, ob die aktuelle Lizenz eine Demolizenz ist. Schreibgeschützt.

string **Error**

Wenn die aktuelle Lizenz nicht gültig ist, zeigt diese Zeichenfolge den Grund dafür an. Schreibgeschützt.

bool **InvoiceEmailTestPerInstance**

Zeigt an, ob das erweiterte E-Mail-Testen pro Instanz in Rechnung gestellt wird.

string **LicenseID**

Die Lizenznummer der Installation. Schreibgeschützt.

bool **LicenseValid**

Zeigt an, ob die Lizenz gültig ist.

string **OwnerName**

Der Name des Lizenzinhabers. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessage-Schnittstelle

Beschreibung

„IMHMessage“ bietet Zugriff auf eine Nachricht.

Anwendung

„IMHMessage“ wird in Plug-Ins als Schnittstelle für eine *Nachricht* verwendet. Eine Nachricht stellt eine einzelne Nachricht dar.

Übergeordnete Schnittstelle

„IMHMessage“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ContentText**

Enthält den Nachrichteninhalt, wenn er als Zeichenfolge dargestellt wird.

Hinweis: Eine Nachricht verwendet nur eine der Eigenschaften *ContentText, ContentBinary* und *MessageUNC.*

string **Context**

Gibt den Kontext zurück, in dem die Nachricht erstellt wurde. Schreibgeschützt.

Hinweis: Dieser Wert ist nur gültig, wenn die Nachricht mit *einem* Kunden verbunden ist. Wenn die Nachricht mit *mehreren* Kunden verbunden ist, verwenden Sie die Eigenschaft *Customers*.

string **ControlParams**

Bietet Zugriff auf die Steuerparameter der Nachricht. Diese Parameter sind in einer Zeichenfolge im Format „<Parametername>=<Parameterwert>“ gespeichert. Die Parameter werden durch Zeilenvorschub getrennt.

string **ControlParamValue**[*string* ParamName]

Bietet Zugriff auf einen einzelnen Steuerungsparameter.

IMHCustomerContainer Customers

Bietet Zugriff auf ein Objekt, das die mit einer Nachricht verbundenen Kunden darstellt.

Hinweis: Diese Eigenschaft ist nur gültig, wenn die Nachricht mit *mehreren* Kunden verbun-

variant **ContentBinary**

Enthält den Nachrichteninhalt, wenn er als Byte-Array dargestellt wird.

Hinweis: Eine Nachricht verwendet nur eine der Eigenschaften *ContentText, ContentBinary* und *MessageUNC.*

den ist. Wenn die Nachricht nur mit *einem* Kunden verbunden ist, verwenden Sie die Eigenschaften *CustomerID* und *Context*.

string **CustomerID**

Gibt die ID des Kunden zurück, mit dem die Nachricht verbunden ist. Schreibgeschützt.

Hinweis: Dieser Wert ist nur gültig, wenn die Nachricht mit *einem* Kunden verbunden ist. Wenn die Nachricht mit *mehreren* Kunden verbunden ist, verwenden Sie die Eigenschaft *Customers*.

IMHMessageAttachmentList ExtraAttachments

Bietet Zugriff auf eine Liste von zusätzlichen Anhängen, die gespeichert und mit der Nachricht verbunden wird, wenn die Nachricht selbst in Dialogue Server gespeichert wird (z. B. wenn *PostMessage(...)* aufgerufen wird).

bool **HasExtraAttachments**

Zeigt an, ob eine Nachricht zusätzliche Anhänge besitzt. Wird in der Eigenschaft *ExtraAttachments* gespeichert.

string **MessageIdentifier**

Die Nachricht enthält eine eindeutige Zeichenfolge (GUID), welche die Nachricht identifiziert.

Hinweis: Die Werte von *MessageIdentifier* werden automatisch von Dialogue Server während der Nachrichtenerstellung generiert. Dies passiert bevor die Nachricht zusammengefügt und in der Datenbank gespeichert wird.

int64 **MessageLogID**

Die interne ID der Nachricht (Datenbankspalte MESSAGE_LOG.ML_ID).

string **MessageUNC**

Enthält den vollständigen Pfad und Dateinamen der Nachricht, falls sie als Datei gespeichert ist.

Hinweis: Eine Nachricht verwendet nur eine der Eigenschaften *ContentText, ContentBinary* und *MessageUNC.*

string **PublicURL**

Enthält eine URL zum Anzeigen der Nachricht durch die Anwendung „Customer Web Access“. Die URL basiert auf dem Wert von *MessageIdentifier.* Schreibgeschützt.

Hinweis: Nur URLs zu E-Mail-Nachrichten werden in der aktuellen Version unterstützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageAssembleInfo-Schnittstelle

Beschreibung

„IMHMessageAssembleInfo“ bietet Zugriff auf ein Objekt, das Informationen enthält, die für das Zusammenfügen einer Nachricht benötigt werden.

Anwendung

„IMHMessageAttachment“ wird in Nachrichten-Plug-Ins verwendet, welche die optionale Schnittstelle **IMHCreateMessagePlugin3** implementieren.

Übergeordnete Schnittstelle

„IMHMessageAssembleInfo“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

variant **CustomValue1**

Ein Wert für die freie Verwendung durch das Nachrichten-Plug-In, um Werte zu speichern. Das Plug-In verwendet diese Eigenschaft typischerweise, um Daten zu speichern, wenn das Objekt *AssembleInfo* beim Aufruf der Methode *PrepareAssembleInfo(...)* vorbereitet wird. Das Plug-In kann die Daten wiederverwenden, wenn *AssembleMessage(...)* zu einem späteren Zeitpunkt aufgerufen wird. Weitere Informationen finden Sie unter **IMHCreateMessagePlugin3**.

variant **CustomValue2**

Gleiche Verwendung und gleicher Zweck wie *CustomValue1*.

variant **CustomValue3**

Gleiche Verwendung und gleicher Zweck wie *CustomValue1*.

bool **IsContentBinary**

Zeigt an, ob der Nachrichteninhalt im Binärformat ist. Schreibgeschützt.

integer **MessageBundleID**

Die interne ID des Nachrichtenpakets (Datenbankspalte MESSAGE_BUNDLE_LOG.MBL_ID). Schreibgeschützt.

string **MessageFileExtension**

Die Erweiterung der Dateien dieses Nachrichtentyps (z. B. htm oder txt). Schreibgeschützt.

bool **OnePerCustomer**

Zeigt an, ob ein Nachrichtenpaket eine Nachricht pro Kunden oder eine Nachricht für alle Kunden enthält. Schreibgeschützt.

variant **UnassembledContentBinary**

Der gemeinsame binäre Inhalt für Pakete mit nicht zusammengefügtem Inhalt. Schreibgeschützt.

string **UnassembledContentText**

Der gemeinsame Textinhalt für Pakete mit nicht zusammengefügtem Inhalt. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageAttachment-Schnittstelle

Beschreibung

„IMHMessageAttachment“ bietet Zugriff auf ein Objekt, das einen Nachrichtenanhang darstellt.

Anwendung

„IMHMessageAttachment“ wird in Plug-Ins verwendet, um mit Nachrichten verbundene Anhänge zu verwalten.

Übergeordnete Schnittstelle

„IMHMessageAttachment“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

variant **ContentBinary**	*string* **FileUNC**	*bool* **IsCachedAttachment**
Enthält den Anhangsinhalt, falls dieser im Arbeitsspeicher zwischengespeichert ist.	Enthält den vollen Pfad und Dateinamen des Anhangs, falls dieser als Datei gespeichert ist (nicht im Arbeitsspeicher zwischengespeichert).	Zeigt an, ob der Anhang selbst im Arbeitsspeicher zwischengespeichert oder als Datei gespeichert ist. Falls *true* angezeigt wird, ist der Anhang in der Eigenschaft *ContentBinary* gespeichert. Anderenfalls ist er in der Datei gespeichert, die angegeben wird durch: *FileUNC.*

Methoden

Die folgenden Methoden werden unterstützt:

void* LoadFromFile**(***string FileUNC)

Lädt einen Anhang von der in *FileUNC* angegebenen Datei und speichert ihn im Arbeitsspeicher.

void* SaveToFile**(***string FileUNC)

Speichert eine Kopie des Anhangs in der Datei, die angegeben wird durch: *FileUNC. SaveToFile(...)* arbeitet unabhängig davon, ob der Anhang im Arbeitsspeicher zwischengespeichert ist.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageAttachmentList-Schnittstelle

Beschreibung

„IMHMessageAttachmentList“ bietet Zugriff auf eine Sammlung von Nachrichtenanhängen.

Anwendung

„IMHMessageAttachmentList“ wird in Plug-Ins verwendet, um mit Nachrichten verbundene Anhänge zu verwalten.

Übergeordnete Schnittstelle

„IMHMessageAttachmentList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

IMHMessageAttachment Attachment[*integer* Index]

Bietet Zugriff auf einen einzelnen Nachrichtenanhang.

Index ist der Index des Anhangs. Der Indexwert „0“ stellt den ersten Anhang dar.

integer **Count**

Gibt die Anzahl der Anhänge in dieser Sammlung zurück.

Methoden

Die folgenden Methoden werden unterstützt:

IMHMessageAttachment AddAttachment( )

Erstellt einen neues Anhangsobjekt und fügt es der Sammlung hinzu.

***void* Clear**( )

Löscht die Sammlung von Nachrichtenanhängen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageBundle-Schnittstelle

Beschreibung

„IMHMessageBundle“ bietet Zugriff auf Nachrichtenpakete.

Anwendung

„IMHMessageBundle“ wird in Plug-Ins als Schnittstelle für ein *Nachrichtenpaket* verwendet. Wenn Nachrichten erstellt werden, wird ein Nachrichtenpaket erstellt, das eine Liste der erstellten Nachrichten enthält.

Übergeordnete Schnittstelle

„IMHMessageBundle“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Gibt die Anzahl der Nachrichten im Paket zurück.

integer **CustDomainID**

Die ID der aktuellen Kundendomäne. Nachrichten werden immer im Kontext *einer* Kundendomäne erstellt. Schreibgeschützt.

bool **IsContentUnassembeled**

Zeigt an, ob der gespeicherte Inhalt für jede Nachricht im Paket zusammengesetzt ist. Wenn er es nicht ist, müssen die Nachrichten zusammengesetzt werden, bevor diese angezeigt, versendet oder anders verwendet werden können. Wird nur bei Nachrichten angewandt, deren Inhalt in der Datenbank gespeichert ist. *IsContentUnassembled* ist eine schreibgeschützte Eigenschaft.

IMHMessage Items[*integer* Index]

Bietet Zugriff auf eine einzelne Nachricht im Paket.

Index ist der Index der Nachricht. Der Indexwert „0“ stellt die erste Nachricht im Paket dar.

integer **MessageBundleID**

Die interne ID des Nachrichtenpakets (Datenbankspalte MESSAGE_BUNDLE_LOG.MBL_ID).

bool **StorageOptionUseTempTables**

Bestimmt, ob der Server temporäre Datenbanktabellen verwenden kann, um die Nachrichtenerstellung zu optimieren.

variant **UnassembledContentBinary**

Der gemeinsame binäre Inhalt für Pakete mit nicht zusammengefügtem Inhalt. Wird nur verwendet, wenn *IsContentUnassembled* auf *true* gesetzt ist.

string **UnassembledContentText**

Der gemeinsame Textinhalt für Pakete mit nicht zusammengefügtem Inhalt. Wird nur verwendet, wenn *IsContentUnassembled* auf *true* gesetzt ist.

IMHUnmergedMessage UnmergedMessage

Bietet Zugriff auf ein Objekt, das die Vorlage oder Basisnachricht darstellt, die verwendet wird oder wurde, um die Nachricht zu erstellen.

Methoden

Die folgenden Methoden werden unterstützt:

void **AddMessage**(**IMHMessage** Message)

Fügt ein Nachrichtenobjekt zum Paket hinzu.

void **Clear**( )

Löscht alle Nachrichtenobjekte im Paket aus dem Arbeitsspeicher.

Hinweis: Wenn die Nachrichten im Paket zuvor in der Datenbank durch Aufrufen von „IMHCustomer.PostMessage(...)“ oder „IMHCustomerContainer.PostMessages(...)“ gespeichert wurden, sind die Informationen in der Datenbank nicht betroffen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHMessageUtils-Schnittstelle

Beschreibung

IMHMessageUtils bietet Zugriff auf Nachrichten und Nachrichtenvorlagen.

Anwendung

„IMHMessageUtils“ wird in Plug-Ins verwendet, um mit Nachrichten und Nachrichtenvorlagen zu arbeiten.

Übergeordnete Schnittstelle

„IMHMessageUtils“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHUnmergedMessage GetMessageTemplate(*integer* BaseMessageID)

Gibt eine Nachrichtenvorlage entsprechend einer Nachrichtenvorlage in Visual Dialogue zurück.

BaseMessageID ist die eindeutige ID der Nachrichtenvorlage. Diese ID findet man in der Datenbanktabelle DOC_BASE_MESSAGE in der Spalte DBM_ID.

IMHUnmergedMessage GetUnmergedMessage(*string* MessageTypeName, *integer* CustDomainID)

Gibt ein Vorlagenobjekt eines angegebenen, in Dialogue Admin definierten Nachrichtentyps zurück. Beachten Sie, dass das zurückgegebene Objekt nicht auf einer speziellen Mastervorlage oder Nachrichtenvorlage basiert.

MessageTypeName ist der technische Name des Nachrichtentyps, z. B. EMAIL_HTML.

CustDomainID ist die eindeutige ID der Kundendomäne, auf die sich mit der zurückgegebenen Vorlage erstellte Nachrichten beziehen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHOutputChannelInfo-Schnittstelle

Beschreibung

IMHOutputChannelInfo bietet Zugriff auf Informationen über den Ausgangskanal, wenn Nachrichten versendet werden.

Anwendung

„IMHOutputChannelInfo“ wird beim Versenden von Nachrichten in Ausgangskanal-Plug-Ins verwendet.

Übergeordnete Schnittstelle

„IMHOutputChannelInfo“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ControlParamValue**[*string* ParamName]

Gibt einen Steuerparameterwert zurück, der für den Ausgangskanal in Dialogue Admin eingerichtet wurde.

ParamName ist der Name des Parameters, wie er vom Ausgangskanal-Plug-In in der Funktion *GetControlParamDefs* definiert ist.

Hinweis: Über diese Eigenschaft sind nur solche Steuerparameterwerte zugänglich, die definiert sind mit: *IsMergeParam=False.* Parameter mit *IsMergeParam=True* werden in der Vorlage oder Basisnachricht eingerichtet und es sollte nur mit der Schnittstelle *IMHMessage* auf sie zugegriffen werden.

Methoden

Die folgenden Methoden werden unterstützt:

void **ReportProgressStatus**(*string* StatusText, *integer* PercentCompleted)

Es werden Berichte an den Dialogue Server-Status und Fortschrittsmeldungen gesendet.

StatusText ist eine Zeichenfolge, die den gegenwärtigen Status beschreibt.

PercentCompleted ist die Prozentangabe, die den Fortschritt des Sendevorgangs angibt. Gültig sind Werte zwischen 0 und 100.

Hinweis: *StatusText* unterstützt die Übersetzung (Lokalisierung). Der Übersetzungsmechanismus ist ein interner Mechanismus, der für die Lokalisierung der Standard-Skript-Plug-Ins verwen-

det wird und nicht für benutzerdefinierte oder Add-On-Plug-Ins anwendbar ist. Siehe auch die *Translate(..)*- und *TranslateAndFormat(..)*-Methoden von **IMHDialogServer**.

void **ReportProgressStatusFormat**(*string* StatusText, *array* Params, *integer* PercentCompleted)

Es werden Berichte an den Dialogue Server-Status und Fortschrittsmeldungen gesendet. Wie *ReportProgressStatus(...)* weiter oben, jedoch wird hier die Formatierung von *StatusText* unterstützt.

StatusText ist eine Zeichenfolge, die den gegenwärtigen Status beschreibt.

Params ist ein Array von Werten, das während des Formatierens in die Zeichenfolge (*Msg*) eingefügt wird.

PercentCompleted ist die Prozentangabe, die den Fortschritt des Sendevorgangs angibt. Gültig sind Werte zwischen 0 und 100.

Formatierungsbeispiel (JScript):

ReportProgressStatusFormat("Aktueller Status: Element {0} von {1} wird bearbeitet", new array(5, 10), 50);

Die Textplatzhalter {0} und {1} werden mit Werten aus dem Array *Params* (Formatierung) ersetzt. Dies führt zu folgendem Ergebnis:

„Aktueller Status: Element 5 von 10 wird bearbeitet“

Beispiele

Es sind keine Beispiele verfügbar.

IMHOutputChannelPlugin-Schnittstelle

Beschreibung

IMHOutputChannelPlugin enthält Methoden, die von Ausgangskanal-Plug-Ins implementiert werden.

COM-basierte Plug-Ins implementieren diese und die übergeordneten Schnittstellen. Skript-basierte Plug-Ins implementieren nur die Methoden dieser Schnittstelle.

Anwendung

Die Logik, die eine Nachricht elektronisch durch einen Kanal versendet, wird durch „IMHOutputChannelPlugin“ implementiert.

Übergeordnete Schnittstelle

„IMHOutputChannelPlugin“ erbt von der Basisschnittstelle aller Plug-Ins: **IMHPlugin**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **GetControlParamDefs**(**IMHControlParamDefs** ControlParamDefs)

„GetControlParamDefs“ wird von Dialogue Server aufgerufen, um die Definition eines Parameters zu erhalten, der vom Plug-In für das Versenden von Nachrichten benötigt wird.

ControlParamDefs ist ein Objekt, das den zu definierenden Parametersatz darstellt.

void **SendMessages**(**IMHOutputChannelInfo** OutputChannelInfo, **IMHChannelMessageContainer** OutputMessages)

„SendMessages“ implementiert die Logik, die eigentlich die Nachricht versendet. Dialogue Server ruft „SendMessages“ auf, wenn ein Satz von Nachrichten versendet werden soll.

OutputChannelInfo ist ein Objekt, das Informationen enthält, die vom Plug-In zum Versenden von Nachrichten benötigt wird.

OutputMessages ist ein Containerobjekt, das eine Liste der Nachrichten enthält. Das Plug-In schleift typischerweise alle Nachrichten in *OutputMessages* durch und versucht sie zu versenden. Das Plug-In gibt für jede Nachricht unter Verwendung der Methoden von *OutputMessages* den Status an Dialogue Server zurück.

Beispiele

Es sind keine Beispiele verfügbar.

IMHParticipant-Schnittstelle

Beschreibung

IMHParticipant bietet Zugriff auf einen einzelnen Dialogteilnehmer.

Anwendung

„IMHParticipant“ wird in Verzweigungs-Plug-Ins verwendet.

Übergeordnete Schnittstelle

„IMHParticipant“ erbt von und erweitert die Schnittstelle **IMHCustomer**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

int64 **ParticipantID**

Bietet Zugriff auf die eindeutige ID des Teilnehmers. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **Accept**( )

Verschiebt den Teilnehmer in die Empfängergruppe der aktuellen Verzweigung in der Operation, die ausgeführt wird.

void **Inactivate**( )

Deaktiviert den Teilnehmer. Deaktivierte Teilnehmer sind nicht länger aktive Dialogteilnehmer. Der Verlauf der Teilnehmer wird dennoch weiterhin in Dialogue Database gespeichert.

Beispiele

Das folgende Beispiel zeigt, wie man Kunden traversiert:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   //Open the container
   Participants.Open("mh_customer_id;Address");

   while (! Participants.MemberEof) {

     //Do something with the customer
     var Street = Participants.Participant.FieldValue("Address.StreetName");

     Participants.Participant.SetCategory("NEWSLETTER");

     //Move the participant to the to-group of the branch
     Participants.Participant.Accept();

     //Move to next participants
     Participants.NextMember();
   }
   ..............
}
```

IMHParticipantContainer-Schnittstelle

Beschreibung

IMHParticipantContainer bietet Zugriff auf einen Satz oder eine Auswahl von Dialogteilnehmern.

Anwendung

„IMHParticipantContainer“ wird in Verzweigungs-Plug-Ins verwendet und bietet Zugriff auf die von der Operation gehandhabten Teilnehmer.

Übergeordnete Schnittstelle

„IMHParticipantContainer“ erbt von und erweitert **IMHCustomerContainer**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

IMHParticipant Participant

Gibt einen Bezug zu einem Objekt zurück, das den aktuellen Teilnehmer darstellt. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

integer **AcceptAll**( )

Verschiebt alle Teilnehmer im Container in die Empfängergruppe der Verzweigung und gibt die Anzahl der verschobenen Teilnehmer zurück.

Die Teilnehmer, die von der Von-Gruppe in die Empfängergruppe verschoben werden, können mit den Methoden *SelectXXX* oder *ExcludeXXX* vor dem Aufrufen von „AcceptAll“ genauer ausgewählt werden.

integer **AcceptBySQL**(*string* SQLName, *array* Params)

Bietet Zugriff, um Teilnehmer manuell von der Von-Gruppe in die Empfängergruppe mittels einer benutzerdefinierten SQL-Anweisung zu verschieben. Die SQL-Anweisung muss der Regel folgen, die gilt, wenn die Tabelle DLG_PARTICIPANT aktualisiert wird.

SQLName ist der Name einer SQL-Anweisung, die in Dialogue Admin im *SQL-Repository* gespeichert ist.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

Hinweis: Das Feld DLG_PARTICIPANT.DLG_INTERNAL_ID muss gleich dem Wert von *Participants.InternalID* in der angegebenen SQL-Anweisung sein.

Warnung: Die Funktion *Erweiterte Ausführung* in Visual Dialogue funktioniert nicht bei der Verwendung von *AcceptBySQL*. Dies liegt daran, dass die Methode eine Verknüpfung um das Standardoperations-Framework bereitstellt.

integer **AcceptBySQLRaw**(*string* SQLStatement, *array* Params, *string* ConnectionName)

AcceptBySQLRaw bewirkt dasselbe wie *AcceptBySQL*. Allerdings wird eine interne SQL-Anweisung verwendet, anstelle des Bezugs zu einer Anweisung, die in Dialogue Admin im SQL-Repository gespeichert ist.

„SQLStatement“ ist die auszuführende SQL.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ConnectionName ist ein Name, der sich auf eine *sekundäre Datenbank* bezieht, die in Dialogue Admin definiert wird. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

void **ExcludeBySQL**(*string* SQLName, *array* Params, *bool* ContextIncluded)

Initialisiert den Container, um alle Teilnehmer der Von-Gruppe auszuwählen, außer diejenigen, die von der angegebenen SQL zurückgegeben werden.

SQLName ist der Name einer SQL-SELECT-Anweisung, die in Dialogue Admin im *SQL-Repository* gespeichert ist. Die SELECT-Anweisung muss eine Spalte namens *mh_customer_id* zurückgeben, welche die Kunden-IDs enthält. Optional kann eine Spalte namens *mh_context* zurückgegeben werden, die Kontextwerte enthält.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ContextIncluded gibt an, ob die SELECT-Anweisung die Spalte *mh_context* zurückgibt.

void **ExcludeBySQLRaw**(*string* SQLStatement, *array* Params, *bool* ContextIncluded, *string* ConnectionName)

Initialisiert den Container, um alle Teilnehmer der Von-Gruppe auszuwählen, außer diejenigen, die von der angegebenen SQL zurückgegeben werden.

SQLStatement ist eine SQL-SELECT-Anweisung. Die SELECT-Anweisung muss eine Spalte namens *mh_customer_id* zurückgeben, welche die Kunden-IDs enthält. Optional kann eine Spalte namens *mh_context* zurückgegeben werden, die Kontextwerte enthält.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ContextIncluded gibt an, ob die SELECT-Anweisung die Spalte *mh_context* zurückgibt.

ConnectionName ist ein Name, der sich auf eine *sekundäre Datenbank* bezieht, die in Dialogue Admin definiert wird. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

integer **InactivateAll**( )

Deaktiviert alle Teilnehmer im Container und gibt die Anzahl der deaktivierten Teilnehmer zurück. Deaktivierte Teilnehmer sind nicht länger aktive Dialogteilnehmer. Der Verlauf der Teilnehmer wird dennoch weiterhin in Dialogue Database gespeichert.

Die deaktivierten Teilnehmer können durch die Methoden *SelectXXX* oder *ExcludeXXX* vor dem Aufrufen von „InactivateAll“ genauer ausgewählt werden.

void **SelectByExpression**(*string* FilterExpression, *string* ContextExpression)

Initialisiert den Container, um alle Teilnehmer in der Von-Gruppe auszuwählen, die den angegebenen Ausdrücken entsprechen.

FilterExpression ist der Ausdruck, der die Kundenfilterkriterien angibt.

ContextExpression ist ein optionaler Ausdruck, der den Kontextwert der Teilnehmer definiert. Setzen Sie *ContextExpression* auf eine leere Zeichenfolge, wenn kein Kontext verwendet wird.

void **SelectBySelection**(*integer* SelectionID)

Initialisiert den Container, um alle Teilnehmer in der angegebenen Auswahl auszuwählen.

SelectionID ist die eindeutige ID der Auswahl.

void **SelectBySQL**(*string* SQLName, *array* Params, *bool* ContextIncluded)

Initialisiert den Container, um alle Teilnehmer in der Von-Gruppe auszuwählen, die auch vom angegebenen SQL zurückgegeben werden.

SQLName ist der Name einer SQL-SELECT-Anweisung, die in Dialogue Admin im *SQL-Repository* gespeichert ist. Die SELECT-Anweisung muss eine Spalte namens *mh_customer_id* zurückgeben,

welche die Kunden-IDs enthält. Optional kann eine Spalte namens *mh_context* zurückgegeben werden, die Kontextwerte enthält.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ContextIncluded gibt an, ob die SELECT-Anweisung die Spalte *mh_context* zurückgibt.

void **SelectBySQLRaw**(*string* SQLStatement, *array* Params, *bool* ContextIncluded, *string* Connection-Name)

Initialisiert den Container, um alle Teilnehmer in der Von-Gruppe auszuwählen, die auch vom angegebenen SQL zurückgegeben werden.

SQLStatement ist eine SQL-SELECT-Anweisung. Die SELECT-Anweisung muss eine Spalte namens *mh_customer_id* zurückgeben, welche die Kunden-IDs enthält. Optional kann eine Spalte namens *mh_context* zurückgegeben werden, die Kontextwerte enthält.

Params ist ein Array, das Parameterwerte enthält, die mit der SQL-Anweisung verbunden sein sollen.

ContextIncluded gibt an, ob die SELECT-Anweisung die Spalte *mh_context* zurückgibt.

ConnectionName ist ein Name, der sich auf eine *sekundäre Datenbank* bezieht, die in Dialogue Admin definiert wird. Setzen Sie „ConnectionName“ auf *null*, um die Standardverbindung zu verwenden.

void **SelectSingle**(*string* CustomerID, *string* Context)

Wählt einen einzelnen Teilnehmer der Von-Gruppe aus. Rufen Sie die Methode erneut auf, um mehrere Teilnehmer auszuwählen.

CustomerID ist die Kunden-ID.

Context ist der Teilnehmerkontext. Wenn kein Kontext verwendet wird, setzen Sie *Context* auf „0“.

Beispiele

Das folgende Beispiel zeigt, wie man Teilnehmer traversiert:

```
function ExecuteBranch(BranchInfo, Participants)
{
   ..............
   //Open the container
   Participants.Open("mh_customer_id;Address");

   while (! Participants.MemberEof) {

     //Do something with the participant / customer

     //Move to next participants
     Participants.NextMember();
   }
   ..............

}
```

IMHPlugin-Schnittstelle

Beschreibung

IMHPlugin ist die Basisschnittstelle aller COM-basierten Plug-Ins. Alle Plug-Ins, die als COM-Objekte geschrieben sind, *müssen* diese Schnittstelle implementieren. Skript-basierte Plug-Ins sollten die Implementierung der Methoden in dieser Schnittstelle ignorieren.

Anwendung

„IMHPlugin“ enthält eine Methode, die von Dialogue Server bei der Initialisierung des Plug-Ins aufgerufen wird.

Übergeordnete Schnittstelle

„IMHPlugin“ erbt von der Basis-COM-Schnittstelle *IUnkown.*

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **InitializePlugin**(**IMHDialogServerServices** DialogServer)

InitializePlugin wird durch Dialogue Server aufgerufen, nachdem das COM-Objekt (das Plug-In) geladen wurde, und bevor alle anderen Methoden des Plug-Ins aufgerufen werden.

DialogServer ist eine Referenz zu einem Objekt, das verschiedene Eigenschaften und Methoden unterstützt. Das Plug-In sollte diese Referenz für die spätere Verwendung in anderen Methoden behalten.

Hinweis: Das Objekt „DialogServer“ wird automatisch für Skript-implementierte Plug-Ins verfügbar gemacht.

Beispiele

Es sind keine Beispiele verfügbar.

IMHQryAlternative-Schnittstelle

Beschreibung

IMHQryAlternative ermöglicht den Zugriff auf Informationen über eine einzelne Alternative in einer Frage in einem Fragebogen (siehe **IMHQryQuestion** und **IMHQuestionnaire**).

Anwendung

„IMHQryAlternative“ wird in Plug-Ins zum Zugriff auf Fragebögen verwendet.

Übergeordnete Schnittstelle

„IMHQryAlternative“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Caption**

Die Beschriftung oder der Titel der Alternative. Schreibgeschützt.

string **DataType**

Der Datentyp von Antworten dieser Alternative. Die verschiedenen Datentypen finden Sie unter **Questionnaire Data Types**.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Alternative. Schreibgeschützt. Die entsprechende Datenbankspalte ist QRY_ALTERNATIVE.QQA_ID.

string **Key**

Der Schlüssel der Alternative. Alternativenschlüssel sind innerhalb einer Frage eindeutig. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHQryQuestion-Schnittstelle

Beschreibung

IMHQryQuestion ermöglicht den Zugriff auf Informationen über eine einzelne Frage im Fragebogen (siehe **IMHQuestionnaire**).

Anwendung

„IMHQryQuestion“ wird in Plug-Ins zum Zugriff auf Fragebögen verwendet.

Übergeordnete Schnittstelle

„IMHQryQuestion“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **AlternativeCount**

Die Gesamtanzahl der Alternativen für die Frage.

IMHQryAlternative Alternatives[*variant* KeyOrIndex]

Ermöglicht den Zugriff auf eine Alternative dieser Frage. Alternativen werden zum Beispiel bei Mehrfachoptionsfragen verwendet.

KeyOrIndex ist entweder ein Alternativenschlüssel (string) oder der Index (integer) der Alternative. Die Indexierung hat die Basis null („0“) und beginnt entsprechend mit „0“ für die erste Alternative.

string **Caption**

Die Beschriftung oder der Titel der Frage. Schreibgeschützt.

string **DataType**

Der Datentyp der Antworten zu diesem Fragebogen. Die verschiedenen Datentypen finden Sie unter **Questionnaire Data Types**.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Frage. Schreibgeschützt. Die entsprechende Datenbankspalte ist QRY_QUESTION.QQU_ID.

string **Key**

Der Schlüssel der Frage. Die Frageschlüssel sind innerhalb eines Fragebogens eindeutig. Schreibgeschützt.

string **QuestionType**

Der Typ der Frage. Schreibgeschützt.

QuestionType kann die folgenden Werte haben:

Wert	Beschreibung
simple	Normale Fragen wie Text- oder Datumswerte.
multi_options	Der Antwortende wählt mindestens eine Option aus einer Reihe von Alternativen aus.
Bewertung	Eine Reihe von Alternativen wird vom Antwortenden in eine Rangfolge gebracht.
Bewertung	Der Antwortende bewertet eine Reihe von auf sich beziehenden Alternativen mithilfe von Bewertungen.

boolean **SingleChoice**

Der Wert *true* (wahr) oder *false* (falsch) legt fest, ob ein oder mehrere Werte in einer Frage des Typs „multi_options“ ausgewählt werden können.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHQrySection-Schnittstelle

Beschreibung

IMHQrySection ermöglicht den Zugriff auf Informationen über einen einzelnen Abschnitt in einem Fragebogen (siehe **IMHQuestionnaire**).

Anwendung

„IMHQrySection“ wird in Plug-Ins zum Zugriff auf Fragebögen verwendet.

Übergeordnete Schnittstelle

„IMHQrySection“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Caption**

Die Beschriftung oder der Titel des Abschnitts. Schreibgeschützt.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID des Abschnitts. Schreibgeschützt. Die entsprechende Datenbankspalte ist QRY_SECTION.QS_ID.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHQuestionnaire-Schnittstelle

Beschreibung

IMHQuestionnaire ermöglicht den Zugriff auf Informationen über einen Fragebogen.

Anwendung

„IMHQuestionnaire“ wird in Plug-Ins zum Zugriff auf Fragebögen verwendet.

Übergeordnete Schnittstelle

„IMHQuestionnaire“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

variant **CustDomainID**

Die eindeutige ID der Kundendomänen, für die der Fragebogen verwendet wird.

Hinweis: Dieser Wert wird ausschließlich dann verwendet, wenn *SurveyType* auf „identified“ festgelegt und die Verwendung des Fragebogens auf die angegebene Kundendomäne beschränkt wurde.

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID des Fragebogens. Schreibgeschützt.

string **Name**

Der Name des Fragebogens. Schreibgeschützt.

integer **QuestionCount**

Die Gesamtanzahl der Fragen im Fragebogen.

IMHQryQuestion Questions[*variant* KeyOrIndex]

Ermöglicht den Zugriff auf eine Frage im Fragebogen.

KeyOrIndex ist entweder ein Frageschlüssel (string) oder der Index (integer) der Frage. Die Indexierung hat die Basis null („0“) und beginnt entsprechend mit „0“ für die erste Frage.

string **SurveyType**

Der Umfragetyp, für den dieser Fragebogen verwendet wird. Schreibgeschützt.

SurveyType kann die folgenden Werte haben:

Wert	Beschreibung

identified	Ausschließlich bekannte Kunden können den Fragebogen ausfüllen. Antwortformulare gehören IMMER zu einem Kunden in einer Kundendomäne.
anonym	Ausschließlich anonyme Antwortende können den Fragebogen ausfüllen. Antwortformulare gehören NICHT zu einem Kunden in einer Kundendomäne.
mixed	Sowohl Umfragen vom Typ „identified“ als auch „anonymous“ werden unterstützt.

integer **SectionCount**

Die Gesamtanzahl der Abschnitte im Fragebogen.

IMHQrySection Sections[*integer* Index]

Ermöglicht den Zugriff auf einen Abschnitt im Fragebogen.

Index ist der Index des Abschnitts, beginnend mit „0“ für den ersten Abschnitt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHQuestionnaireUtils-Schnittstelle

Beschreibung

IMHQuestionnaireUtils bietet Zugriff auf Fragebögen und Antwortformulare.

Anwendung

„IMHQuestionnaireUtils“ wird im Plug-Ins verwendet, um auf Fragebögen und Antwortformulare zuzugreifen.

Übergeordnete Schnittstelle

„IMHQuestionnaireUtils“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHAnswerFormList GetAnonymousAnswerForms(*integer* QuestionnaireID, *bool* OnlyUnprocessed, *integer* MaxCount)

Gibt eine Liste von anonymen Antwortformularen zurück.

QuestionnaireID ist die eindeutige ID des Fragebogens.

OnlyUnprocessed gibt an, ob nur Antwortformulare zurückgegeben werden, die nicht als verarbeitet markiert sind. Ein Antwortformular kann als verarbeitet markiert werden, indem die Methode *MarkAsProcessed( )* aufgerufen wird (siehe **IMHAnswerForm**).

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

IMHAnswerForm GetAnswerForm(*int64* AnswerFormID)

Gibt ein Objekt zurück, das ein Antwortformular darstellt.

AnswerFormID ist die eindeutige ID des Antwortformulars, entsprechend der Datenbankspalte QRY_ANSWER_FORM.QAF_ID.

IMHAnswerForm GetAnswerFormByCustomer(*integer Questionnaire*ID, *integer* CustDomainID, *string* CustomerID, *string* Context)

Gibt ein Objekt zurück, das ein Antwortformular darstellt, welches mit dem angegebenen Kunden und Kontext verbunden ist.

QuestionnaireID ist die eindeutige ID des Fragebogens.

CustDomainID ist die ID der Kundendomäne, zu welcher der Kunde gehört.

CustomerID ist die eindeutige ID des Kunden, der den Fragebogen beantwortet hat.

Context ist der Kontext, in dem der Kunde den Fragebogen beantwortet hat. „Context“ wird in einem Dialog verwendet und dient der Unterscheidung zwischen mehreren Teilnehmern mit derselben Kunden-ID.

IMHAnswerFormList GetAnswerFormsBySQL(*string* SQLName, *array* Params, *integer* MaxCount)

Gibt eine Liste von Antwortformularen entsprechend der Antwortformular-IDs zurück, die von der angegebenen SQL-SELECT-Anweisung zurückgegeben werden.

SQLName ist der Name des Elements im SQL-Repository. Dies definiert die zu verwendende SELECT-Anweisung. Die SELECT-Anweisung sollte Zeilen zurückgeben, welche die Antwortformular-ID in der ersten Spalte enthalten.

Params ist ein Array, das Parameterwerte für die SQL-Anweisung enthält (Datenbank-Host-Variablen).

„MaxCount“ zeigt die maximale Anzahl abzurufender Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare entsprechend der SQL-Anweisung zurück.

IMHAnswerFormList GetAnswerFormsBySQLRaw(*string* SQLStatement, *array* Params, *string* ConnectionName, *integer* MaxCount)

Bewirkt dasselbe wie *GetAnswerFormBySQL( ).* Allerdings wird die SQL-Anweisung während des Methodenaufrufs und nicht durch einen Verweis zum SQL-Repository angegeben.

SQLStatement ist die zu verwendende SQL-SELECT-Anweisung.

IMHAnswerFormList GetIdentifiedAnswerForms(*integer* QuestionnaireID, *bool* OnlyUnprocessed, *integer* MaxCount)

Gibt eine Liste der identifizierten Antwortformulare zurück.

QuestionnaireID ist die eindeutige ID des Fragebogens.

OnlyUnprocessed gibt an, ob nur Antwortformulare zurückgegeben werden, die nicht als verarbeitet markiert sind. Ein Antwortformular kann als verarbeitet markiert werden, indem die Methode *MarkAsProcessed( )* aufgerufen wird (siehe **IMHAnswerForm**).

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

IMHAnswerFormList GetIdentifiedAnswerFormsEx(*integer* QuestionnaireID, *bool* OnlyUnprocessed, *bool* IncludeComplete, *bool* IncludeUncomplete, *integer* MaxCount)

Gibt eine Liste der identifizierten Antwortformulare zurück.

QuestionnaireID ist die eindeutige ID des Fragebogens.

OnlyUnprocessed gibt an, ob nur Antwortformulare zurückgegeben werden, die nicht als verarbeitet markiert sind. Ein Antwortformular kann als verarbeitet markiert werden, indem die Methode *MarkAsProcessed( )* aufgerufen wird (siehe **IMHAnswerForm**).

IncludeComplete legt fest, dass ausschließlich als abgeschlossen markierte Antwortformulare zurückgegeben werden.

IncludeIncomplete legt fest, dass ausschließlich nicht als abgeschlossen markierte Antwortformulare zurückgegeben werden.

Hinweis: Sie können nicht gleichzeitig *IncludeComplete* und *IncludeIncomplete* auf **false** setzen.

MaxCount zeigt die maximale Anzahl der abzurufenden Antwortformulare an. Das Setzen von „MaxCount“ auf „-1“ gibt alle Antwortformulare zurück, welche die weiteren Kriterien erfüllen.

IMHQuestionnaire GetQuestionnaire(*integer* QuestionnaireID)

Gibt ein Objekt zurück, das einen angegebenen Fragebogen darstellt.

QuestionnaireID ist die eindeutige ID des zurückzugebenden Fragebogens.

Beispiele

Es sind keine Beispiele verfügbar.

IMHReportEngine-Schnittstelle

Beschreibung

Die *IMHReportEngine*-Schnittstelle unterstützt die Berichtsfunktionalität im Dialogue Server.

Anwendung

„IMHReportEngine“ wird vom **DialogServer** -Objekt durch seine *ReportEngine*-Eigenschaft unterstützt.

Übergeordnete Schnittstelle

„IMHReportEngine“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Es werden keine Eigenschaften unterstützt.

Methoden

Die folgenden Methoden werden unterstützt:

integer **GenerateArchivedReport**(*integer* ReportTemplateID, *string* Params, *string* ReportName)

Erzeugt einen Bericht und speichert das Ergebnis im Berichtsarchiv der Datenbank. Es wird die eindeutige ID des gespeicherten archivierten Berichts zurückgegeben.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall.

ReportName ist der Name des Berichts im Archiv.

Hinweis: Verwenden Sie *GetArchivedReport(..)*, um das Ergebnis eines erzeugten Berichts in einem bestimmten Dokumentformat zu erhalten.

string **GenerateArchivedReportEx**(*integer* ReportTemplateID, **IMHReportParameterList** Params, *string* ReportName)

Bewirkt das Gleiche wie *GenerateArchivedReport*(..), außer dass *Params* einen anderen Typ hat.

Params ist eine Liste von Parametern mit den zur Generierung des Berichts verwendeten Parameterwerten und -operatoren.

Hinweis: Rufen Sie *GetParameterList(..)* aus **IMHReportTemplate** auf, um eine Parameterliste für eine bestimmte Vorlage zu erhalten.

string **GenerateReport**(*integer* ReportTemplateID, *string* Params, *integer* FormatIndex)

Erzeugt einen Bericht im festgelegten Format und gibt das Ergebnis als XML-Dokument zurück.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

string **GenerateReportEx**(*integer* ReportTemplateID, **IMHReportParameterList** Params, *integer* FormatIndex)

Bewirkt das Gleiche wie *GenerateReport(..)*, außer dass *Params* einen anderen Typ hat.

Params ist eine Liste von Parametern mit den zur Generierung des Berichts verwendeten Parameterwerten und -operatoren.

Hinweis: Rufen Sie *GetParameterList(..)* aus **IMHReportTemplate** auf, um eine Parameterliste für eine bestimmte Vorlage zu erhalten.

void **GenerateReportUNC**(*integer* ReportTemplateID, *string* Params, *integer* FormatIndex, *string* ReportUNC)

Erzeugt einen Bericht im festgelegten Format und speichert diesen in einer Datei.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

ReportUNC ist der vollständige Pfad und Dateiname zur Zielberichtsdatei.

Hinweis: Auf dem Datenträger wird lediglich die Hauptberichtsdatei gespeichert, d. h. Berichtsformate, die mehrere Dateien erzeugen, werden nicht unterstützt.

string **GenerateReportUNCEx**(*integer* ReportTemplateID, **IMHReportParameterList** Params, *integer* FormatIndex, *string* ReportUNC)

Bewirkt das Gleiche wie *GenerateReportUNC(..)*, außer dass *Params* einen anderen Typ hat.

Params ist eine Liste von Parametern mit den zur Generierung des Berichts verwendeten Parameterwerten und -operatoren.

Hinweis: Rufen Sie *GetParameterList(..)* aus **IMHReportTemplate** auf, um eine Parameterliste für eine bestimmte Vorlage zu erhalten.

string **GetArchivedReport**(*integer* ReportArchiveID, *integer* FormatIndex)

Gibt einen archivierten Bericht im festgelegten Format als XML-Dokument zurück.

ReportArchiveID ist die eindeutige ID des archivierten Berichts.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

void **GetArchivedReportUNC**(*integer* ReportArchiveID, *integer* FormatIndex, *string* ReportUNC)

Speichert einen archivierten Bericht mit dem festgelegten Format in einer Datei.

ReportArchiveID ist die eindeutige ID des archivierten Berichts.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des Berichts festgelegt wird. Die Format-Zahlenwerte können durch das Aufrufen von *GetReportDeviceTypes(..)* abgerufen werden.

ReportUNC ist der vollständige Pfad und Dateiname zur Zielberichtsdatei.

Hinweis: Auf dem Datenträger wird lediglich die Hauptberichtsdatei gespeichert, d. h. Berichtsformate, die mehrere Dateien erzeugen, werden nicht unterstützt.

string **GetReportParamSchema**(*string* ServerSession)

Gibt das XML-Schema zurück, das zur Festlegung der Parameterwerte in den Aufrufen an „GenerateReport(..)“ und „GenerateArchivedReport(..)“ verwendet wird.

Beispiele

Es sind keine Beispiele verfügbar.

IMHReportFormat-Schnittstelle

Beschreibung

IMHReportFormat ermöglicht den Zugriff auf Informationen über das Berichtsausgabeformat.

Anwendung

„IMHReportFormat“ wird von Plug-Ins bei der Generierung von und Arbeiten mit Berichten verwendet.

Übergeordnete Schnittstelle

„IMHReportFormat“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Description**

Description ist eine Zeichenfolge, die das Format beschreibt.
. Schreibgeschützt.

string **FileExtention**

FileExtention ist die für Dateien des Berichtsformats verwendete Erweiterung. Schreibgeschützt.

integer **FormatIndex**

Ermöglicht den Zugriff auf eine Ganzzahl, die das Berichtsformat identifiziert. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHReportParameter-Schnittstelle

Beschreibung

IMHReportParameter ermöglicht den Zugriff auf einen Berichtsparameter.

Anwendung

„IMHReportParameter“ wird in Plug-Ins zum Generieren von und Arbeiten mit Berichten verwendet.

Übergeordnete Schnittstelle

„IMHReportParameter“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Datatype**

Der Datentyp des Parameters. Eine Liste von Datentypen finden Sie unter **Datentypen**. Schreibgeschützt.

string **Operator**

Operator ist der im Parameter verwendete Vergleichsoperator, z. B. <=.

string **ParamName**

ParamName ist der Name des Parameters. Schreibgeschützt.

bool **Required**

Gibt an, ob für den Parameter ein Wert erforderlich ist. Wenn nicht, kann *Value* auf *null* festgelegt werden. Schreibgeschützt.

string **SystemType**

SystemType enthält den Namen des Systemparametertyps, wenn er damit verwandt ist. Anderenfalls ist *SystemType* eine leere Zeichenfolge. Schreibgeschützt.

variant **Value**

Value ist der dem Parameter zugeordnete Wert. Der Datentyp von *Value* sollte mit dem von „Datatype“ festgelegten Typ übereinstimmen.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHReportParameterList-Schnittstelle

Beschreibung

IMHReportParameterList ermöglicht den Zugriff auf eine Liste von Berichtsparametern.

Anwendung

„IMHReportParameterList“ wird in Plug-Ins zum Generieren von und Arbeiten mit Berichten verwendet.

Übergeordnete Schnittstelle

„IMHReportParameterList“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **Count**

Count legt die Zahl der Parameter in der Liste fest. Schreibgeschützt.

IMHReportParameter Parameter[*variant* NameOrIndex]

Ermöglicht den Zugriff auf einen einzelnen Parameter in der Liste.

NameOrIndex ist entweder der Name eines Parameters (string) oder der Index (integer) des Parameters in der Liste. Die Indexierung hat die Basis null („0“) und beginnt entsprechend mit „0“ für den ersten Parameter.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHReportTemplate-Schnittstelle

Beschreibung

IMHReportTemplate ermöglicht den Zugriff auf Informationen über eine Berichtsvorlage.

Anwendung

„IMHReportTemplate“ wird in Plug-Ins zum Zugriff auf Berichtsvorlagen verwendet.

Übergeordnete Schnittstelle

„IMHReportTemplate“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Berichtsvorlage. Schreibgeschützt.

string **Name**

Name ist der Name der Berichtsvorlage. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

IMHReportParameterList GetParameterList( )

Gibt eine Liste der für die Vorlage definierten Berichtsparameter zurück. Die zurückgegebene Liste kann zur Festlegung von Parameterwerten und -operatoren sowie als ein Parameter beim Generieren von Berichten mithilfe der **IMHReportEngine**-Methoden verwendet werden.

string **GetUniqueFileUNC**(*string* NamePrefix, *integer* FormatIndex)

Gibt einen eindeutigen, auf einem Präfix und einem bestimmten Berichtsformat basierenden Dateinamen zurück. Der Dateiname ist eindeutig in Bezug auf die Dialogue Server-Instanz.

NamePrefix ist eine Zeichenfolge, die den ersten Teil des zu generierenden Dateinamens enthält. Beispielsweise ist dies der Name der Berichtsvorlage.

FormatIndex ist eine Zahl, die das Dokumentformat dieses Berichts angibt. Diese wird zur Bestimmung des Erweiterungsteils des zurückgegebenen Dateinamens verwendet. Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

Beispiele

Es sind keine Beispiele verfügbar.

IMHSelection-Schnittstelle

Beschreibung

IMHSelection ermöglicht den Zugriff auf Informationen über eine Auswahl.

Anwendung

„IMHSelection“ wird in Plug-Ins zum Zugriff auf Auswahlen verwendet.

Übergeordnete Schnittstelle

„IMHSelection“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der Auswahl. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHSQLDef-Schnittstelle

Beschreibung

IMHSQLDef ermöglicht den Zugriff auf eine im SQL-Repository in Dialogue Admin gespeicherte SQL-Anweisung.

Anwendung

„IMHSQLDef“ wird in Plug-Ins als eine Schnittstelle zu einer gespeicherten SQL-Anweisung verwendet.

Übergeordnete Schnittstelle

„IMHSQLDef“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

integer **ID**

Ermöglicht den Zugriff auf die eindeutige ID der SQL-Anweisung. Schreibgeschützt.

string **Name**

Name ist der Name der SQL-Anweisung im SQL-Repository. Schreibgeschützt.

string **SQLStatement**

Ermöglicht den Zugriff auf die gespeicherte SQL-Anweisung als eine Zeichenfolge. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

integer **Execute**(*array* Params)

Führt die SQL aus und gibt die Anzahl der betroffenen Zeilen zurück.

Params ist ein Array, das die in die SQL-Anweisung einzubindenden Parameterwerte enthält.

object **Open**(*array* Params)

Öffnet und gibt einen ADO-Datensatz mithilfe der gespeicherten SQL-Anweisung zurück.

Params ist ein Array, das die in die SQL-Anweisung einzubindenden Parameterwerte enthält.

Beispiele

Es sind keine Beispiele verfügbar.

IMHSystemUser-Schnittstelle

Beschreibung

IMHSystemUser bietet Zugriff auf den aktuellen Systembenutzer. Das heißt, der Benutzerkontext, in dem das Plug-In ausgeführt wird.

Anwendung

„IMHSystemUser“ wird in Plug-Ins verwendet, um auf Eigenschaften des Systembenutzers zuzugreifen, z. B. Benutzername oder E-Mail-Adresse.

Übergeordnete Schnittstelle

„IMHSystemUser“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **Cellular**

Die Mobiltelefonnummer des Benutzers. Dies ist ein optionales Feld bei der Definition neuer Systembenutzer in Dialogue Admin. Schreibgeschützt.

string Description

Eine optionale Beschreibung des Benutzers. Schreibgeschützt.

string **DisplayName**

Der vollständige Name des Benutzers. Wird in der Regel zu Anzeigezwecken verwendet. Schreibgeschützt.

string **Email**

Die E-Mail-Adresse des Benutzers. Dies ist ein optionales Feld bei der Definition neuer Systembenutzer in Dialogue Admin. Schreibgeschützt.

string **UserName**

Der Benutzername (Anmeldungsname) des Benutzers. Schreibgeschützt.

Methoden

Es werden keine Methoden unterstützt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHUnmergedMessage-Schnittstelle

Beschreibung

IMHUnmergedMessage ermöglicht den Zugriff auf eine Nachrichtenvorlage bzw. eine Basisnachricht.

Anwendung

„IMHUnmergedMessage“ wird in einem Plug-In als eine Schnittstelle zu einer Vorlage bzw. einer Basisnachricht verwendet. Es stellt den von einem *Nachrichten-Plug-In* zur Erstellung von Nachrichten benötigten Zugriff auf die Vorlagendaten zur Verfügung.

Übergeordnete Schnittstelle

„IMHUnmergedMessage“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

string **ActivityDesc**

Stellt bei der Erstellung einer Nachricht den Zugriff zum Lesen oder Einrichten der Beschreibung der zu übertragenden Aktivität bereit.

integer **ActivityTypeID**

Die ID des zu übertragenen Aktivitätstyps. Schreibgeschützt.

string **ActivityTypeName**

Der technische Name des zu übertragenen Aktivitätstyps. Schreibgeschützt.

string **ActualFileEncoding**

ActualFileEncoding ist der Name der beim Speichern der Nachrichten in Dateien verwendeten Codierung. Diese Einstellung wird nur auf Textdateien angewendet.

bool **AllowOptimizeStorage**

Gibt an, ob die Speicherung von aus dieser Vorlage erstellten Nachrichten optimiert werden kann. Bei ***false*** wird der endgültige Inhalt der Nachricht ebenfalls gespeichert. Diese Option steht nur dann zur Verfügung, wenn der Nachrichteninhalt in der Datenbank gespeichert wird.

integer **BaseMessageID**

Die ID der in Visual Dialogue definierten Nachrichtenvorlage (oder der Basisnachricht). Dieser Wert ist *null*, wenn das aktuelle Vorlagenobjekt nicht zu einer solchen Nachrichtenvorlage gehört. Schreibgeschützt.

string **ControlParams**

Ermöglicht den Zugriff auf die festgelegten Steuerungsparameter. Diese Parameter sind in einer Zeichenfolge im Format „<Parametername>=<Parameterwert>“ gespeichert. Die Parameter werden durch Zeilenvorschub getrennt.

string **ControlParamValue**[*string* ParamName]

Bietet Zugriff auf einen einzelnen Steuerungsparameter.

integer **CustDomainID**

Die ID der zur Nachrichtenvorlage gehörigen Kundendomäne. Schreibgeschützt.

string **DataFields**

DataFields ist eine durch Semikola getrennte Liste von Kundendatengruppen und -feldern (aus der Definition der Kundendomäne), die zur Erstellung der Nachricht benötigt werden.

string **DefaultFileEncoding**

DefaultFileEncoding ist der für den aktuellen Nachrichtentyp verwendete Name der Dateicodierung entsprechend der Konfiguration in Dialogue Admin. Diese Einstellung wird nur auf Textdateien angewendet. Schreibgeschützt.

bool **IsContentBinary**

Gibt an, ob der Nachrichteninhalt im Binärformat erstellt wurde. Das bedeutet, dass dieser nicht als eine Zeichenfolge in Bezug auf Speicherung und Datenaustausch repräsentiert werden kann, es sei denn, es wird eine Binärcodierung angewendet. *IsContentBinary* wird in Dialogue Admin für jeden definierten Nachrichtentyp eingerichtet.

string **MessageFileExtension**

Die Erweiterung der aus dieser Nachrichtenvorlage erstellten Nachrichtendateien (z. B. HTM oder TXT). Schreibgeschützt.

string **MessageName**

Ermöglicht den Zugriff zum Lesen oder Festlegen des Namens der zu erstellenenden Nachricht.

integer **MessageTypeID**

Die ID des verwendeten Nachrichtentyps. Schreibgeschützt.

string **MessageTypeName**

Der technische Name des verwendeten Nachrichtentyps. Schreibgeschützt.

bool **OnePerCustomer**

Ein boolescher Wert, der angibt, ob eine Nachricht pro Kunde (z. B. eine E-Mail) oder eine *einzige* Nachricht, die alle Kunden enthält (z. B. eine Exportdatei), erstellt werden soll.

integer **OutputChannelID**

Die ID des Ausgangskanals, der nach der Erstellung zum Versenden der Nachrichten eingerichtet wird. Wenn kein Ausgangskanal definiert wurde, ist der Wert *NULL*. Schreibgeschützt.

variant **PluginDefID**

Die ID (Ganzzahl) des *Nachrichten-Plug-Ins*, das zur Erstellung von Nachrichten verwendet wird. Der Wert von *PluginDefID* ist NULL, wenn diesem Nachrichtentyp kein Nachrichten-Plug-In zugordnet wurde. Schreibgeschützt. *PluginDefID* wird vom System intern verwendet.

integer **Priority**

Priority steuert die Priorität der Nachricht beim elektronischen Versenden von Nachrichten, z. B. bei E-Mails oder SMS-Nachrichten. *Priority* kann die folgenden fünf möglichen Werte haben:

Wert	Wertname	Beschreibung
-2	Niedrig	Die niedrigste Priorität. Diese Nachrichten werden versendet, nachdem alle anderen Nachrichten in der Sendewarteschlange verarbeitet wurden.
-1	Medium low	Mittelniedrige Priorität.
0	Normal	Die standardmäßige Nachrichtenpriorität.
1	Medium high	Mittelhohe Priorität.
2	Hoch	Die höchste Priorität. Diese Nachrichten werden vor allen anderen Nachrichten in die Sendewarteschlange gesendet.

bool **StoreAsFile**

Ein boolescher Wert, der angibt, ob die Nachrichten als Dateien oder in Dialogue Database gespeichert werden sollen. Nicht alle *Nachrichten-Plug-Ins* unterstützen beide Modi.

string **SQLConnectionID**

Falls *UsesSQL* den Wert „*true“ hat, enthält SqlConnectionID* die ID der Datenbankverbindung, die für die Ausführung des in *SqlStatement* gespeicherten Befehls verwendet wird. Schreibgeschützt.

string **SQLStatement**

Falls *UsesSQL* den Wert „true“ hat, enthält *SQLStatement* die als Quelle verwendete SELECT-Anweisung zum Zusammenführen von Nachrichten. Schreibgeschützt.

string **TargetPathUNC**

Wenn die Nachricht als Datei gespeichert wird (z. B. als eine DOC-Datei), verweist *TargetPathUNC* auf den Netzwerkspeicherort, an dem die Nachricht gespeichert wurde. Dieses Verzeichnis ist auch der Basisspeicherort, der zum Speichern von zur Nachricht gehörigen Anhängen (z. B. E-Mail-Anhänge) verwendet wird.

string **TemplateContent**

Ermöglicht den Zugriff auf den Inhalt einer textbasierten (nicht binären) Nachrichtenvorlage bzw. Basisnachricht.

string **TemplateUNC**

Die UNC mit dem vollständigen Pfad und Dateinamen der Vorlagen- bzw. Basisnachrichtendatei, z. B. ein MS Word-Vorlagendokument.

bool **UseOutbox**

Gibt an, ob Nachrichten nach der Erstellung zum *Postausgang* gesendet werden sollen.

Warnung: Wenn *UseOutbox* auf *false* (FALSCH) eingestellt wurde, werden elektronische Nachrichten, wie z. B. SMS- und E-Mail-Nachrichten, nach der Erstellung automatisch in die Versandwarteschlange aufgenommen.

bool **UseSQL**

Gibt an, ob eine benutzerdefinierte SQL-Anweisung zur Verwendung als Datenquelle zum Mischen von Nachrichten aus dieser Vorlage definiert ist. Eine solche benutzerdefinierte SQL-Anweisung kann in Visual Dialogue beim Definieren der Nachrichtenvorlage definiert werden. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

string **GetNextFilename**( )

Gibt den vollständigen zu verwendenden Pfaddateinamen zum Speichern der nächsten zu erstellenden Nachricht zurück. Mithilfe dieser Methode wird ein eindeutiger Dateiname mit einem korrekten Pfad gemäß der vom Dialogue Server verwendeten Ordnerstruktur sichergestellt.

Beispiele

Es sind keine Beispiele verfügbar.

IMHWebPublicFile-Schnittstelle

Beschreibung

IMHWebPublicFile bietet Zugriff auf Informationen über eine einzelne Datei, die im Internet veröffentlicht wurde.

Anwendung

„IMHWebPublicFile“ wird verwendet, wenn mit veröffentlichten Dateien gearbeitet wird, indem Methoden des Objekts **DialogServer** verwendet werden.

Übergeordnete Schnittstelle

„IMHWebPublicFile“ erbt von der Basisschnittstelle **IMHCustomPluginServices**.

Eigenschaften

Die folgenden Eigenschaften werden unterstützt:

variant **ContentBinary**

Enthält den Inhalt der Datei, der als Byte-Array dargestellt wird. Schreibgeschützt.

string **ContentID**

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert. Schreibgeschützt.

string **Description**

Description ist eine Zeichenfolge, welche die Datei beschreibt. Schreibgeschützt.

string **FileName**

FileName ist der Name der Datei und der Erweiterungsteile. Pfadinformationen sind davon ausgeschlossen. *Beispiel:* meinlogo.jpg.

FileName ist schreibgeschützt.

float **Size**

Gibt die Größe der Datei als Byte-Anzahl zurück. Schreibgeschützt.

string **URL**

Gibt die URL zurück, die verwendet wird, um über das Internet auf die Datei zuzugreifen. Die URL verweist auf die Anwendung „Web Utilities“. Schreibgeschützt.

Methoden

Die folgenden Methoden werden unterstützt:

void **SaveToFile**(*string FileUNC*)

Speichert den Inhalt der Datei in einer vom Benutzer angegebenen Datei. Der Inhalt der Eigenschaft *ContentBinary* wird gespeichert.

FileUNC ist der vollständige Pfad und Dateiname der zu erstellenden Datei.

Beispiele

Es sind keine Beispiele verfügbar.

Kapitel 11

Dialogue Server-API

In diesem Abschnitt:

- Dialogue Server-API 250
- Activity-API 251
- Dialogue-API 255
- Customer-API 263
- Generic-API 271
- Message-API 274
- Emarketing Mail-API 287
- Quest-API 289
- Report-API 293
- Selection-API 298
- System-API 301
- Telemarketing-API 305
- Web Utilities-API 308

Dialogue Server-API

Einleitung

Der Dialogue Server verfügt über eine öffentliche API: die *Dialogue Server-API*. Auf diese API kann sowohl über *Microsoft COM* als auch mittels SOAP über *WebServices* zugegriffen werden.

Die verschiedenen APIs

Die folgenden WEB-Dienste und COM-Komponenten sind definiert; jede/r mit einem Satz passender SOAP-Operationen. Die meisten dieser SOAP-Operationen können auch mit HTTP POST- und GET-Methoden verwendet werden, siehe dazu die Beschreibung der einzelnen Operationen.

- **System-API**

  Diese API beinhaltet Operationen auf Systemebene.

- **Customer-API**

  Diese API ermöglicht den Zugriff auf und die Bearbeitung von Kundeninformationen.

- **Dialogue-API**

  Diese API ermöglicht den Zugriff auf und die Bearbeitung von Dialogen und Teilnehmern.

- **Emarketing Mail-API**

  Diese API wird beim Arbeiten mit Nachrichten-Designer-Vorlagen verwendet.

- **Message-API**

  Diese API ermöglicht den Zugriff auf, sowie die Erstellung und Aktualisierung von Kundennachrichten.

- **Quest-API**

  Diese API ermöglicht den Zugriff auf und die Aktualisierung von Fragebögen und Antwortformularen.

- **Activity-API**

  Diese API ermöglicht den Zugriff auf und die Bearbeitung von Aktivitäten und Aufgaben.

- **Telemarketing-API**

  Diese API ermöglicht den Zugriff auf und die Bearbeitung von Telemarketing-Projekten und -Teilnehmern. Zugang zu Telemarketing-Statistiken einbezogen.

- **Report-API**

  Diese API ermöglicht den Zugriff auf und die Generierung von Berichten.

- **Web Utilities-API**

  Eine API, um auf die Funktionen von Webdienstprogrammen, die E-Mail- und Verknüpfungsnachverfolgung beinhalten, sowie auf Dateien zuzugreifen, die im Internet veröffentlicht wurden.

- **Generic-API**

Diese API ermöglicht den Zugriff auf und die Bearbeitung von generischen Objekten. Beispielsweise kann mit SQL-Befehlen auf Datenbanken zugegriffen und in Dialogue Admin Datensätze definiert oder ein Dialogue Server-Plug-In ausgeführt werden.

- **Selection-API**

  Diese API ermöglicht den Zugriff auf und die Verwaltung von Auswahlen.

Der Parameter „ServerSession“

Den meisten Methoden in der Dialogue Server-API wird der Parameter „ServerSession“ übergeben. Dieser Parameter ist eine Zeichenfolge, welche die aktuelle Serversitzung identifiziert. Diesen Sitzungsschlüssel erhält man von der Methode *Login* der System-API mit gültigem Benutzernamen und Kennwort.

Testen der API

Automatisch erstellte Webseiten, welche die SOAP-Operationen und die HTTP GET- und POST-Methoden beschreiben, stehen über die WebServices-Anwendung selbst zur Verfügung. Diese Seiten bieten auch Testmechanismen für die Operationen. Um diese Seiten aufzurufen, besuchen Sie:

http://<Dialogue Server-API Webserver>/<API Installationsverzeichnis>

Beispiel: http://MyWebServer/MHDialogServerAPI

Activity-API

Übersicht

Die Activity-API stellt eine Reihe von Methoden zum Zugriff auf Daten zur Verfügung, die mit Aufgaben in Verbindung stehen.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHActivityAPI* implementiert die Activity-API über die Schnittstelle *IMHActivityAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

void **DeleteActivity**(*string* ServerSession, *int64* ActivityID)
Löscht die angegebene Aktivität.
ActivityID ist die eindeutige ID der Aktivität.
string **GetActivity**(*string* ServerSession, *int64* ActivityID)
Gibt ein XML-Dokument mit Aktivitätsinformationen zurück.
ActivityID ist die eindeutige ID der Aktivität.

string **GetActivitySchema**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das nur den Teil des Schemas enthält, der zurückgegeben wurde von: *GetActivity.*

string **GetActivityTypes**(*string* ServerSession)

Gibt ein XML-Dokument mit allen in Dialogue Admin definierten Aktivitätstypen zurück.

string **GetTaskFUOptions**(*string* ServerSession)

Gibt ein XML-Dokument mit einer Liste von häufig verwendeten Optionen zum Einstellen von Datum und Uhrzeit der Nachverfolgung einer Aufgabe zurück.

Hinweis: *GetTaskFUOptions* wird von Customer View verwendet, und die zurückgegebenen Werte werden physikalisch in der Dialogue Server-Datenbank in der Datenbanktabelle TASK_FU_OPTION gespeichert.

string **GetTasksByUser**(*string* ServerSession, *datetime* FUIntervalBegin, *bool* FUIntervalBeginSet, Set *datetime* FUIntervalEnd, *bool FUIntervalEndSet, bool* IncludeUndone*, bool* IncludeDone, *integer* MaxCount)

Gibt ein XML-Dokument mit einer Liste von Aufgaben zurück, die dem aktuellen Benutzer zur Nachverfolgung zugewiesen wurden. Die zurückgegebenen Aufgaben entsprechen den angegebenen Parametern.

FUIntervalBegin ist das Startdatum (sowie die Uhrzeit) des Nachverfolgungsintervalls (für das Aufgaben zurückgegeben werden). Wenn *FUIntervalBeginSet* auf *false* anstatt auf *true* festgelegt wird, hat das zur Folge, dass der Wert von *FUIntervalBegin* nicht berücksichtigt wird.

FUIntervalEnd ist das Enddatum (sowie die Uhrzeit) des Nachverfolgungsintervalls (für das Aufgaben zurückgegeben werden). Wenn *FUIntervalEndSet* auf *false* anstatt auf *true* festgelegt wird, hat das zur Folge, dass der Wert von *FUIntervalEnd* nicht berücksichtigt wird.

IncludeUndone legt fest, ob noch nicht abgeschlossene Aufgaben eingeschlossen werden.

IncludeDone legt fest, ob abgeschlossene Aufgaben eingeschlossen werden.

MaxCount ist die maximale Anzahl zurückzugebener Aufgaben. Legen Sie MaxCount auf -1 fest, um alle Aufgaben zurückzugeben, die den anderen angegebenen Kriterien entsprechen.

Hinweis: Wenn Sie sowohl *IncludeUndone* als auch *IncludeDone* auf *false* festlegen, wird eine leere Aufgabenliste zurückgegeben.

string **GetTaskSummaryByUser**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das eine numerische Zusammenfassung fälliger Aufgaben enthält. Die Nummern entsprechen Aufgaben, die vom aktuellen Benutzer nachzuverfolgen sind.

string **GetTaskViewIntervals**(*string* ServerSession)

Gibt ein XML-Dokument mit einer Liste von häufig verwendeten Durchsuchungsintervallen beim Anzeigen von Aufgaben zurück. Die Intervalle stehen mit dem Datum und der Uhrzeit der Nachverfolgung einer Aufgabe in Beziehung.

Hinweis: *GetTaskViewIntervals* wird von Customer View verwendet, und die zurückgegebenen Werte werden physikalisch in der Dialogue Server-Datenbank in der Datenbanktabelle TASK_VIEW_INTERVAL gespeichert.

string **GetWorkGroups**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle definierten Aufgabengruppen enthält. Aufgabenarbeitsgruppen werden im Dialogue-Aufgabenorganisator definiert.

int64 **InsertActivity**(*string* ServerSession, *string* ActivityDataXML)

Sendet eine neue Aktivität zum Speichern von Änderungen zum Dialogue Server. Es wird die eindeutige ID der neuen Aktivität zurückgegeben.

ActivityDataXML ist ein XML-Dokument mit geänderten Aktivitätsdaten. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams. Wenn die durch *GetActivitySchema* abgerufene XML in einen ADO-Datensatz geladen wird und dieser Datensatz verändert wird, kann durch ADO.NET automatisch das nach „InsertActivity“ zu sendende DiffGram erstellt werden.

void **MarkTaskAsDone**(*string* ServerSession, *int64* TaskID)

Markiert zum aktuellen Zeitpunkt eine spezielle Aufgabe als abgeschlossen.

TaskID ist die eindeutige ID der Aufgabe. Da eine Aufgabe einen Untertyp einer Aktivität darstellt, ist die TaskID die gleiche wie die ID der entsprechenden Aktivität.

int64 **PostActivity**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *string* ActivityTypeName, *string* ChannelTypeName, *string* Direction, *string* Description, *string* Note, *datetime* Timestamp)

Sendet eine Aktivität und gibt die eindeutige ID der neuen Aktivität zurück. Aktivitäten sind Kundeninteraktionen.

CustDomainID ist die ID, welche die Kundendomäne festlegt, und *CustomerID* ist die ID des mit der neuen Aktivität verbundenen Kunden.

Context ist der Kontextwert des Kunden. Wenn kein Kontext benötigt wird, legen Sie *Context* auf die Zeichenfolge „0“ fest.

ActivityTypeName ist der technische Name des in Dialogue Admin definierten Aktivitätstyps.

ChannelTypeName ist der technische Name des in Dialogue Admin definierten Kanaltyps. Der Kanaltyp beschreibt die Art und Weise der stattgefundenen Kommunikation in der Aktivität oder Interaktion.

Direction beschreibt, ob die Aktivität vom Kunden oder von „uns“ bzw. „dem System“ initiiert wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN": initiiert vom Kunden.
- Direction = "OUT": initiiert von „uns“ bzw. „dem System“.

Description ist ein kurzes Beschreibungsfeld, während es sich bei *Note* um eine unbegrenzt lange Beschreibung handelt.

Timestamp ist der Zeitpunkt, zu dem die Aktivität stattgefunden hat.

int64 **PostTask**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID*, string* Context, *string* ActivityTypeName, *string* ChannelTypeName, *string* Direction*, string* Description, *string* Note, *datetime* Timestamp, *datetime* FollowUpDateTime, *integer* TaskWorkGroupID, *string* FollowUpUserName)

Sendet eine Aufgabe und gibt die eindeutige ID der neuen Aktivität zurück. Aufgaben sind Aktivitäten, die innerhalb einer bestimmten Zeitdauer nachverfolgt werden.

CustDomainID ist die ID, welche die Kundendomäne festlegt, und *CustomerID* ist die ID des mit der neuen Aktivität verbundenen Kunden.

Context ist der Kontextwert des Kunden. Wenn kein Kontext benötigt wird, legen Sie *Context* auf die Zeichenfolge „0“ fest.

ActivityTypeName ist der technische Name des in Dialogue Admin definierten Aktivitätstyps.

ChannelTypeName ist der technische Name des in Dialogue Admin definierten Kanaltyps. Der Kanaltyp beschreibt die Art und Weise der stattgefundenen Kommunikation in der Aktivität oder Interaktion.

Direction beschreibt, ob die Aktivität vom Kunden oder von „uns“ bzw. „dem System“ initiiert wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN": initiiert vom Kunden.
- Direction = "OUT": initiiert von „uns“ bzw. „dem System“.

Description ist ein kurzes Beschreibungsfeld, während es sich bei *Note* um eine unbegrenzt lange Beschreibung handelt.

Timestamp ist der Zeitpunkt, zu dem die Aktivität stattgefunden hat.

FollowUpDateTime ist der Wert für Datum und Uhrzeit, mit dem der Abgabetermin für die Nachverfolgung der Aufgabe festgelegt wird.

TaskWorkGroup ist ein optionaler Wert, der sich auf die für die Nachverfolgung der Aufgabe verantwortliche Arbeitsgruppe bezieht. Legen Sie diesen Parameter auf -1 fest, wenn Sie keine Arbeitsgruppe angeben möchten.

FollowUpUserName ist der Name des für die Nachverfolgung der Aufgabe verantwortlichen Benutzers. Legen Sie diesen Parameter auf eine leere Zeichenfolge fest, wenn Sie keinen verantwortlichen Benutzer angeben möchten.

void **SetTaskReadFlag**(*string* ServerSession, *int64* TaskID, *bool* Read)

Eine Aufgabe verfügt über eine „Lesemarkierung“, die angibt, ob die Aufgabe durch den für die Nachverfolgung verantwortlichen Benutzer geöffnet oder überprüft wurde. Durch diese Methode wird diese Markierung festgelegt bzw. gelöscht.

TaskID ist die eindeutige ID der Aufgabe. Da eine Aufgabe einen Untertyp einer Aktivität darstellt, ist die TaskID die gleiche wie die ID der entsprechenden Aktivität.

Legen Sie *Read* auf *true* fest, um die Aufgabe zum aktuellen Zeitpunkt als „gelesen“ zu markieren. Legen Sie *Read* auf *false* fest, um die „Lesemarkierung“ zu löschen (dadurch wird die Aufgabe als „ungelesen“ markiert).

void **UpdateActivity**(*string* ServerSession, *int64* ActivityID, *string* ActivityDataXML)

Sendet Aktualisierungen einer Aktivität zum Speichern der Änderungen zum Dialogue Server.

ActivityID ist die eindeutige ID der Aktivität.

ActivityDataXML ist ein XML-Dokument mit geänderten Aktivitätsdaten. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams. Wenn die durch *GetActivity* abgerufene XML in einen ADO-Datensatz geladen wird und dieser Datensatz verändert wird, kann durch ADO.NET automatisch das nach *UpdateActivity* zu sendende DiffGram erstellt werden.

Hinweis: *ActivityID* ist theoretisch eine redundante Information. Allerdings ist diese in der aktuellen Version erforderlich, um die Kommunikation von Änderungen zwischen einem Client und dem Dialogue Server zu erleichtern.

Beispiele

Im folgenden Beispiel wird ein XML-Dokument mit einer Übersicht über die Aufgabenzusammenfassung vom Dialogue Server abgerufen. Das Beispiel verwendet JScript.

```
  ..............
  //Connect to the DialogServer
  SystemAPI = new ActiveXObject("MHDialogServer.MHSystemAPI");

  //Log in
  var InstanceName = "MHProduction";
  var SessionKey = SystemAPI.Login(InstanceName, "williams", "mypassword",
"MH Test");
  var ServerSession = SessionKey + "@" + InstanceName;

  //Now get the task summary for the current user
  ActivityAPI = new ActiveXObject("MHDialogServer.MHActivityAPI");
  var TaskSummaryXML = ActivityAPI.GetTaskSummaryByUser(ServerSession);

  //Log out
  SystemAPI.Logout(ServerSession);
  ..............
```

Dialogue-API

Übersicht

Die Dialogue-API stellt eine Reihe von Methoden zum Zugriff auf Daten zur Verfügung, die mit Dialogen und Teilnehmern zusammenhängen.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHDialogAPI* implementiert die Dialogue-API durch die Schnittstelle *IMHDialogAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

void **ClearDialog**(*string* ServerSession, *integer DialogID*, *bool* DeleteActivities, *bool* DeleteMessages)

Leert einen Dialog, indem alle Ausführungsdaten eines Dialogs gelöscht werden.

DialogID ist die ID des gewünschten Dialogs.

DeleteActivities gibt an, ob mit den Dialogteilnehmern verknüpfte Aktivitäten im Dialog gelöscht werden sollen.

„DeleteMessages“ gibt an, ob mit den Dialogteilnehmern verknüpfte Nachrichten im Dialog gelöscht werden sollen.

integer **CreateDialogFromDialogTemplate**(*string* ServerSession, *integer DialogTemplateID*, *string D*ialogSetupUpdateXML, *bool* EnsureUniqueName)

Erstellt einen neuen Dialog auf Grundlage einer Dialogvorlage und aktualisiert die Dialogeinrichtung des neuen Dialogs.

DialogTemplateID ist die ID der Dialogvorlage, auf der der neue Dialog aufbauen soll.

DialogSetupUpdateXML ist ein XML-Dokument mit den aktualisierten Daten der Dialogeinrichtung. Das Schema dieses XML-Dokuments können Sie mit der Methode *GetDialogSetupUpdateSchema(..)* abrufen.

EnsureUniqueName gibt an, ob der Dialogname in der Erstellung verändert werden kann, um seine Eindeutigkeit sicherzustellen.

integer **CreateDialogFromExistingDialog**(*string* ServerSession, *integer DialogID*, *string D*ialogSetupUpdateXML, *bool* EnsureUniqueName)

Erstellt einen neuen Dialog auf Grundlage eines existierenden Dialogs und aktualisiert die Dialogeinrichtung des neuen Dialogs. Der neue Dialog ist dann eine Kopie des existierenden Dialogs. Dabei wird nur die Definition des Dialogs kopiert, nicht dessen Ausführungsdaten.

DialogID ist die ID des existierenden Dialogs, auf der der neue Dialog aufbauen soll.

DialogSetupUpdateXML ist ein XML-Dokument mit den aktualisierten Daten der Dialogeinrichtung. Das Schema dieses XML-Dokuments können Sie mit der Methode *GetDialogSetupUpdateSchema(..)* abrufen.

EnsureUniqueName gibt an, ob der Dialogname in der Erstellung verändert werden kann, um seine Eindeutigkeit sicherzustellen.

void **DeleteDialog**(*string* ServerSession, *integer DialogID*, *bool* DeleteActivities, *bool* DeleteMessages)

Löscht sowohl die Definition als auch die Ausführungsdaten eines Dialogs.

DialogID ist die ID des gewünschten Dialogs.

DeleteActivities gibt an, ob mit den Dialogteilnehmern verknüpfte Aktivitäten im Dialog gelöscht werden sollen.

„DeleteMessages“ gibt an, ob mit den Dialogteilnehmern verknüpfte Nachrichten im Dialog gelöscht werden sollen.

string **EvaluateExpression**(*string* ServerSession, *integer* CustDomainID, *int64 ParticipantID*, *string* Expression)

Gibt ein XML-Dokument mit dem Ergebniswert des angegebenen Ausdrucks zurück.

CustDomainID ist die ID der Kundendomäne; *ParticipantID* ist die ID des Dialogteilnehmers, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck.

bool **EvaluateExpressionBool**(*string* ServerSession, *integer* CustDomainID, *int64 ParticipantID*, *string* Expression)

Wertet einen booleschen Ausdruck aus und gibt das Ergebnis zurück.

CustDomainID ist die ID der Kundendomäne; *ParticipantID* ist die ID des Dialogteilnehmers, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck. Die Angabe eines nicht-booleschen Ausdrucks führt zu einer Fehlermeldung.

string **EvaluateExpressionString**(*string* ServerSession, *integer* CustDomainID, *int64 ParticipantID*, *string* Expression)

Wertet einen Ausdruck aus und gibt das Ergebnis zurück.

CustDomainID ist die ID der Kundendomäne; *ParticipantID* ist die ID des Dialogteilnehmers, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck. Die Angabe eines Ausdrucks, der eine Datengruppe oder ein Array zurückgibt, führt zur Ausgabe einer Ausnahme. Verwenden Sie die Methode *EvaluateExpression* zur Prüfung solcher Ausdrücke.

integer **ExecuteOperation**(*string* ServerSession, *integer* OperationID, *integer* MaxCount)

Führt eine Operation in einem Dialog aus und gibt die Gesamtanzahl der mit dieser Operation bewegten Teilnehmer zurück.

OperationID ist die eindeutige Kennung der Dialogoperation.

MaxCount ist die Maximalzahl der Teilnehmer, die mit dieser Operation bewegt werden können. Setzen Sie diesen Parameter auf -1, um ein unbegrenztes Maximum der Teilnehmerzahl festzulegen.

void **ExecuteOperationAsynchronous**(*string* ServerSession, *integer* OperationID, *integer* MaxCount)

Startet die Ausführung einer Operation in einem Dialog asynchron. Diese Methode wird sofort wieder verlassen, während die Operation im Hintergrund gestartet wird.

OperationID ist die eindeutige Kennung der Dialogoperation.

MaxCount ist die Maximalzahl der Teilnehmer, die mit dieser Operation bewegt werden können. Setzen Sie diesen Parameter auf -1, um ein unbegrenztes Maximum der Teilnehmerzahl festzulegen.

string **GetChannelTypes**(*string* ServerSession)

Gibt ein XML-Dokument mit allen in Dialogue Admin konfigurierten Kanaltypen zurück.

string **GetCustomerContexts**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *integer* MaxCount)

Gibt ein XML-Dokument mit den Kontexten zurück, in denen ein bestimmter Kunde an Dialogen teilnimmt. Ein Kunde kann in einem bestimmten Kontext an einem Dialog teilnehmen. Wenn ein Kunde mehr als einmal am selben Dialog teilnimmt, sind unterschiedliche Kontextwerte erforderlich.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

CustomerID ist die eindeutige ID des Kunden, der als Dialogteilnehmer eingetragen werden soll.

MaxCount ist die Maximalzahl von Kontexten, die abgerufen werden. Die jüngsten Kontexte werden als erstes zurückgegeben. Setzen Sie diesen Parameter auf -1, um alle Kontexte zu erhalten.

string **GetDialog**(*string* ServerSession, *integer* DialogID)

Gibt detaillierte Informationen über eine Dialogdefinition zurück, einschließlich aller Gruppen und Informationen zu „Unterstützte Einrichtung“.

DialogID ist die ID des gewünschten Dialogs.

string **GetDialogDetails**(*string* ServerSession, *integer* DialogID)

Gibt ein XML-Dokument mit detaillierten Informationen über einen Dialog zurück. Dieses Dokument beschreibt die Dialoggestaltung einschließlich einer Liste aller Gruppen im Dialog. Nicht enthalten sind Daten zur Dialogausführung (siehe *GetDialogHistory*).

DialogID ist die ID des gewünschten Dialogs.

Hinweis: Verwenden Sie die Methode „GetDialog()“. Die Methode „GetDialogDetails()“ wird für Rückwärtskompatibilität angeboten.

string **GetDialogHistory**(*string* ServerSession, *integer* DialogID)

Gibt ein XML-Dokument mit detaillierten Ausführungsinformationen eines Dialogs zurück.

DialogID ist die ID des gewünschten Dialogs.

string **GetDialogs**(*string* ServerSession)

Gibt ein XML-Dokument mit allen definierten Dialogen zurück. Die Dialoge im XML-Dokument sind nach Kundendomäne kategorisiert.

string **GetDialogsEx**(*string* ServerSession)

Gibt ein XML-Dokument mit allen definierten Dialogen und erweiterten Informationen zurück.

string **GetDialogTemplate**(*string* ServerSession, *integer* DialogTemplateID)

Gibt detaillierte Informationen über eine Dialogdefinition zurück, einschließlich aller Gruppen und Informationen zu „Unterstützte Einrichtung“.

DialogTemplateID ist die ID des gewünschten Dialogs.

string **GetDialogTemplates**(*string* ServerSession)

Gibt ein XML-Dokument mit allen definierten Dialogvorlagen zurück. Die Vorlagen im XML-Dokument sind nach Kundendomäne kategorisiert.

string **GetDialogSetupUpdateSchema**(*string* ServerSession)

Gibt die zur Aktualisierung der Dialogeinstellungen verwendete XML-Definition zurück.

variant **GetFirstDialogExecution**(*string* ServerSession, *integer* DialogID)

Gibt den Zeitpunkt der ersten Ausführung aller Operationen in einem Dialog zurück. Das Ergebnis ist null, wenn keine Vorgänge ausgeführt wurden.

DialogID ist die ID des gewünschten Dialogs.

integer **GetParticipantCount**(*string* ServerSession, *integer* CustDomainID, *integer* GroupID, *string* FilterExpression)

Gibt die Anzahl Teilnehmer zurück, die die angegebenen Kriterien erfüllen.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

GroupID ist die ID der Dialoggruppe, welche die Teilnehmer enthält.

FilterExpression ist ein Ausdruck, der zur Filterung oder Verfeinerung der Auswahl der gezählten Kunden verwendet wird.

string **GetParticipantHistory**(*string* ServerSession, *int64* ParticipantID)

Gibt ein XML-Dokument mit detaillierten Ausführungsinformationen eines Dialogteilnehmers zurück.

ParticipantID ist die ID des gewünschten Teilnehmers.

string **GetParticipants**(*string* ServerSession, *integer* CustDomainID, *integer* GroupID, *string* FilterExpression, *string* DataFields, *integer* MaxCount)

Gibt ein XML-Dokument mit enthaltenen Teilnehmern und ihren Kundendaten zurück.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

GroupID ist die ID der Dialoggruppe, welche die Teilnehmer enthält.

FilterExpression ist ein Ausdruck, der zur Filterung oder Verfeinerung der gezählten Kunden verwendet wird.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

MaxCount ist die Maximalzahl der zurückgegebenen Kunden. MaxCount auf -1 einstellen, um alle Teilnehmer zurückzugeben.

string **GetParticipantSchema**(*string* ServerSession, *integer* CustDomainID, *string* DataFields)

Gibt ein XML-Dokument zurück, das nur das Schema der Teilnehmerdaten enthält. Dies ist das gleiche XML-Dokument, das auch „GetParticipants“ und „GetSingleParticipant“ zurückgeben, es enthält jedoch keine Daten.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

string **GetParticipantSelection**(*string* ServerSession, *integer* GroupID, *integer* SelectionID, *string* DataFields, *integer* MaxCount)

Gibt ein XML-Dokument mit enthaltenen Teilnehmern und ihren Kundendaten zurück. Die enthaltenen Teilnehmer *müssen* in der angegebenen Dialoggruppe *und* gleichzeitig in der angegebenen Auswahl sein.

GroupID ist die ID der Dialoggruppe, welche die Teilnehmer enthält.

SelectionID ist die ID der Auswahl.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

MaxCount ist die Maximalzahl der zurückgegebenen Kunden. MaxCount auf -1 einstellen, um alle Teilnehmer zurückzugeben.

integer **GetParticipantSelectionCount**(*string* ServerSession, *integer* GroupID, *integer* SelectionID)

Gibt die Anzahl Teilnehmer zurück, die die angegebenen Kriterien erfüllen. Die enthaltenen Teilnehmer *müssen* in der angegebenen Dialoggruppe *und* gleichzeitig in der angegebenen Auswahl sein.

GroupID ist die ID der Dialoggruppe, welche die Teilnehmer enthält.

SelectionID ist die ID der Auswahl.

string **GetParticipantSelectionSorted**(*string* ServerSession, *integer* GroupID, *integer* SelectionID, *string* DataFields, *string* SortFields, *integer* MaxCount)

Wie *GetParticipantSelection(...)*, jedoch kann die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

string **GetParticipantsSorted**(*string* ServerSession, *integer* CustDomainID, *integer* GroupID, *string* FilterExpression, *string* DataFields, *string* SortFields, *integer* MaxCount)

Wie *GetParticipants(...)*, jedoch kann die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

string **GetSingleParticipant**(*string* ServerSession, *int64* ParticipantID, *string* DataFields)

Gibt ein XML-Dokument mit den Kundendaten eines einzelnen Teilnehmers zurück.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

ParticipantID ist die ID des zurückzugebenen Teilnehmers.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

void **InactivateParticipant**(*string* ServerSession, *integer* ParticipantID)

Deaktiviert einen Teilnehmer in einem Dialog. *Deaktivieren* heißt hier, dass der Teilnehmer nicht mehr am Dialog teilnimmt. Der Teilnehmereintrag und der Teilnehmerverlauf bleiben jedoch in Dialogue Database gespeichert.

ParticipantID ist die ID des gewünschten Teilnehmers.

bool **InsertParticipant**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *integer* GroupID)

Fügt einen neuen Teilnehmer in die angegebene Gruppe in einem Dialog ein.

Gibt *true* zurück, wenn der neue Teilnehmer erfolgreich eingefügt wurde. Wenn der Kunde (einschließlich Kontext) bereits Mitglied des Dialogs ist, wird *false* zurückgegeben.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

CustomerID ist die eindeutige ID des Kunden, der als Dialogteilnehmer eingetragen werden soll.

Context ist der Zeichenfolgenkontextwert, der bei einigen Dialogen verwendet wird. Falls der Kontext nicht verwendet wird, setzen Sie diesen Parameter auf den Zeichenfolgenwert „0“.

GroupID ist die ID der Dialoggruppe, in die der neue Teilnehmer eingefügt wird.

Hinweis: Nur Gruppen, die in Visual Dialogue mit *Manuelle Einfügungen von Teilnehmern zulassen* markiert wurden, können beim Aufruf von *InsertParticipant* angegeben werden.

string ***ParticipantLogin***(*string* ServerSession, *integer* CustDomainID, *integer* DialogID, *int64* ParticipantID, *string* Password)

Authentifiziert einen Kunden mit seiner „ParticipantID“ (anstelle seiner *LoginID*, wie es die Funktion *MHCustomerAPI.CustomerLogin* tut). Bei erfolgreicher Authentifizierung wird die Kunden-ID des Kunden zurückgegeben, anderenfalls wird eine Ausnahme ausgegeben.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

DialogID ist die ID des Dialogs, an dem der Kunde teilnimmt. Wenn *DialogID* den Wert -1 hat, werden alle Teilnehmer-IDs akzeptiert, die zu Dialogen gehören, für die Anmeldungen aktiviert sind.

ParticipantID ist die ID des zu authentifizierenden Teilnehmers.

Password ist das Kennwort, das unter der Kundendomäne in Dialogue Admin definiert wurde. *Password* kann jedes Feld aus der Kundendomäne oder ein automatisch generierter Wert sein.

Hinweis: *ParticipantLogin()* funktioniert nur für Dialoge, für welche die Kundenanmeldung aktiviert wurde. Diese können Sie in Visual Dialogue unter „Dialogeigenschaften“ einrichten.

string ***ParticipantLoginEx***(*string* ServerSession, *integer* CustDomainID, *integer* DialogID, *int64 ParticipantID*, *string* Password, *bool* PasswordRequired)

ParticipantLoginEx funktioniert wie *ParticipantLogin*, jedoch wird das Kennwort ausgelassen.

PasswordRequired gibt an, ob der Server das Kennwort prüfen soll oder nicht.

void **PostAnonymousEvent**(*string* ServerSession, *string* EventTypeName*, string* Description)

Sendet ein Systemereignis, das NICHT mit einem bestimmten Kunden verbunden ist**.** Systemereignisse werden in Dialogue Admin definiert. Dialogue Server kann so eingerichtet werden, dass auf unterschiedliche Systemereignisse mit unterschiedlichen Aktivitäten reagiert wird. Beispielsweise kann ein Vorgang in einem oder mehreren Dialogfeldern ausgeführt werden.

EventTypeName ist die technische Bezeichnung des in Dialogue Admin definierten Ereignistyps.

Description ist die optionale Beschreibung eines Ereignisses.

void **PostEvent**(*string* ServerSession, *string* EventTypeName*, string* Description*, integer* CustDomainID, *string* CustomerID*, string* Context)

Sendet ein Systemereignis, das mit einem bestimmten Kunden verbunden ist**.** Systemereignisse werden in Dialogue Admin definiert. Dialogue Server kann so eingerichtet werden, dass auf unterschiedliche Systemereignisse mit unterschiedlichen Aktivitäten reagiert wird. Beispielsweise kann ein Vorgang in einem oder mehreren Dialogfeldern ausgeführt werden.

EventTypeName ist die technische Bezeichnung des in Dialogue Admin definierten Ereignistyps.

Description ist die optionale Beschreibung eines Ereignisses.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des mit dem Ereignis verbundenen Kunden.

Context ist der Kontextwert des Kunden. Wenn kein Kontext benötigt wird, legen Sie *Context* auf die Zeichenfolge „0“ fest.

void **UpdateDialogSetup**(*string* ServerSession, *integer* DialogID, *string* DialogSetupUpdateXML, *bool* EnsureUniqueName)

Aktualisiert die Einrichtungsdaten eines Dialogs.

DialogID ist die ID des Dialogs, der mit veränderten oder neuen Einrichtungsdaten aktualisiert werden soll.

DialogSetupUpdateXML ist ein XML-Dokument mit den aktualisierten Daten der Dialogeinrichtung. Das Schema dieses XML-Dokuments können Sie mit der Methode *GetDialogSetupUpdateSchema(..)* abrufen.

EnsureUniqueName gibt an, ob der Dialogname in der Erstellung verändert werden kann, um seine Eindeutigkeit sicherzustellen.

Beispiele

Im folgenden Beispiel werden Teilnehmerinformationen aus Dialogue Server abgerufen. Das Beispiel verwendet JScript.

```
..............
//Connect to the DialogServer
SystemAPI = new ActiveXObject("MHDialogServer.MHSystemAPI");

//Log in
```

```
   var InstanceName = "MHProduction";
   var SessionKey = SystemAPI.Login(InstanceName, "admin", "ringo1", "MH
Test");
   var ServerSession = SessionKey + "@" + InstanceName;

   //Now get some participants
   DialogAPI = new ActiveXObject("MHDialogServer.MHDialogAPI");
  var ParticipantXML = DialogAPI.GetParticipants(ServerSession, 1000, 5040,
 "", "LastName;FirstName", -1);

   //Log out (we got what we wanted)
   SystemAPI.Logout(ServerSession);
   ..............
```

Customer-API

Übersicht

Die Customer-API bietet eine Reihe von Methoden zum Zugriff auf kundenzugehörige Daten.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHCustomerAPI* implementiert die Customer-API durch die Schnittstelle *IMHCustomerAPI.*

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

bool **CheckExpressionSyntax**(*string* ServerSession, *integer* CustDomainID, *string* Expression)

Prüft die Syntax eines Ausdrucks, indem sie ihn kompiliert und analysiert. Die resultierende XML enthält das Ergebnis einschließlich einer Liste von Syntaxanmerkungen.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

Expression ist der zu prüfende Ausdruck.

string **CheckForDuplicates**(*string* ServerSession, *integer* CustDomainID, *string* CustomerDataXML, *string* DataFields*, integer MaxCount*)

Prüft eine Kundendomäne auf mögliche Duplikate und gibt eine Liste dieser möglichen Duplikate als XML zurück. Die Prüfung findet anhand von Kundeninformationen statt, die der Methode übergeben werden mit dem Parameter: *CustomerDataXML.*

CustDomainID ist die Kennung der Kundendomäne, die auf Duplikate geprüft wird.

CustomerDataXML ist ein XML-Dokument mit Kundendaten. Diese Daten werden als Grundlage der Prüfung verwendet; Dialogue Server wird also die Kundendatenbank nach Einträgen durchsuchen, die diesen Daten entsprechen. Das Format des XML-Dokuments entspricht dem XML-Dokument, das die Methode *GetSingleCustomer(...) zurückgibt.*

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

MaxCount ist die Maximalzahl der zurückgegebenen möglichen Duplikate. Setzen Sie *MaxCount* auf -1, um alle potentiellen Duplikate zu erhalten.

string ***CustomerLogin***(*string* ServerSession, *integer* CustDomainID, *string* LoginID, *string* Password)

Authentifiziert einen Kunden. Bei erfolgreicher Authentifizierung wird die Kunden-ID zurückgeben. Anderenfalls wird eine Ausnahme ausgegeben.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

LoginID und *Password* sind Werte die in der Kundendomäne in Dialogue Admin definiert werden. *LoginID* kann so eingerichtet werden, dass sie einem eindeutigen Feld in der Kundendomäne entspricht, etwa die Kunden-ID oder die E-Mail-Adresse.*Password* kann jedes Feld aus der Kundendomäne oder ein automatisch generierter Wert sein.

Hinweis: Wenn mehr als ein Kunde mit der angegebenen Kunden-ID während der Authentifizierung gefunden wird, wird eine Ausnahme mit einer entsprechenden Fehlermeldung ausgegeben.

string ***CustomerLoginEx***(*string* ServerSession, *integer* CustDomainID, *string* LoginID, *string* Password, *bool* PasswordRequired)

CustomerLoginEx funktioniert wie *CustomerLogin*; das Kennwort wird jedoch ausgelassen.

PasswordRequired gibt an, ob der Server das Kennwort prüfen soll oder nicht.

void **DeleteSingleCustomer**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID)

Löscht einen einzelnen Kunden. Die Informationen über den Kunden werden auch in Dialogue Database entfernt: z. B. die Teilnahme an Dialogen und Antwortformularen sowie Nachrichten.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des zu löschenden Kunden.

Hinweis: Wenn der Kunde in der angegebenen Kundendomäne nicht existiert, wird ein Fehler ausgegeben.

string **EvaluateExpression**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Expression)

Gibt ein XML-Dokument mit dem Ergebniswert des angegebenen Ausdrucks zurück.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des Kunden, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck.

bool **EvaluateExpressionBool**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Expression)

Wertet einen booleschen Ausdruck aus und gibt das Ergebnis zurück.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des Kunden, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck. Die Angabe eines nicht-booleschen Ausdrucks führt zu einer Fehlermeldung.

string **EvaluateExpressionString**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Expression)

Wertet einen Ausdruck aus und gibt das Ergebnis zurück.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des Kunden, für den der Ausdruck evaluiert wird.

Expression ist der auszuwertende Ausdruck. Die Angabe eines Ausdrucks, der eine Datengruppe oder ein Array zurückgibt, führt zur Ausgabe einer Ausnahme. Verwenden Sie die Methode *EvaluateExpression* zur Prüfung solcher Ausdrücke.

string **GetCategories**(*string* ServerSession, *integer* CustDomainID)

Gibt ein XML-Dokument mit allen in Dialogue Admin definierten Kategorien zurück.

string **GetCustDomain**(*string* ServerSession, *integer* CustDomainID)

Gibt ein XML-Dokument zurück, das eine einzelne Kundendomäne beschreibt.

CustDomainID ist die Kennung, welche die Kundendomäne angibt.

Hinweis: Verwenden Sie die Methode *GetCustDomains*, um die Definition aller Kundendomänen zu erhalten.

string **GetCustDomainList**(*string* ServerSession)

Gibt ein XML-Dokument mit einer einfachen Liste aller definierten Kundendomänen zurück.

string **GetCustDomains**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das eine Beschreibung aller definierten Kundendomänen enthält. Deaktivierte Kundendomänen sind nicht enthalten.

Hinweis: Verwenden Sie die Methode *GetCustDomain*, um die Definition einer einzelnen Kundendomäne zu erhalten.

string **GetCustomerByLoginID**(*string* ServerSession, *integer* CustDomainID, *string Login*ID*, string* DataFields)

Gibt ein XML-Dokument mit den Daten eines einzelnen Kunden zurück, der den angegebenen Kriterien entspricht. Die Auswahl des Kunden basiert auf der angegebenen Anmelde-ID. Die Kundenanmeldung muss für die angegebene Kundendomäne in Dialogue Admin aktiviert sein.

CustDomainID ist die ID der Kundendomäne; *LoginID* ist die Anmelde-ID des Kunden, wie in Dialogue Admin definiert.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

integer **GetCustomerCount***(string* ServerSession, *integer* CustDomainID, *string* FilterExpression, *string* ContextExpression)

Gibt die Anzahl Kunden zurück, die die angegebenen Kriterien erfüllen.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

FilterExpression ist ein Ausdruck zur Filterung oder Verfeinerung der Auswahl der gezählten Kunden.

ContextExpression wird, wenn angegeben, den Kunden Kontextwerte hinzufügen, bevor sie gezählt werden. „ContextExpression“ ist ein Ausdruck, der eine Zeichenfolge oder ein Array von Zeichenfolgen zurückgibt. Im Fall eines Arrays von Zeichenfolgen können mehrere Kontextwerte pro Kunde existieren und der Kunde kann mehr als einmal (nämlich einmal pro Kontextwert) gezählt werden.

string **GetCustomers***(string* ServerSession, *integer* CustDomainID, *string* FilterExpression, *string* ContextExpression*, string* DataFields, *integer* MaxCount)

Gibt ein XML-Dokument der Kunden mit Kundendaten zurück, die den angegebenen Kriterien entsprechen.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

FilterExpression ist ein Ausdruck zur Filterung oder Verfeinerung der Auswahl der gezählten Kunden.

ContextExpression wird, wenn angegeben, einen Wert im Feld *mh_context* im zurückgegebenen XML-Dokument einfügen. „ContextExpression“ ist ein Ausdruck, der eine Zeichenfolge oder ein Array von Zeichenfolgen zurückgibt. Im Fall eines Arrays von Zeichenfolgen können mehrere Kontextwerte pro Kunde existieren und der Kunde kann mehrere Male (nämlich einmal pro Kontextwert) gezählt werden.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

MaxCount ist die Maximalzahl der zurückgegebenen Kunden. MaxCount muss auf -1 eingestellt werden, um alle Kunden zurückzugeben, die die anderen angegebenen Kriterien erfüllen.

string **GetCustomerSchema***(string* ServerSession, *integer* CustDomainID, *string* DataFields)

Gibt ein XML-Dokument nur mit dem Schema der Kundendaten zurück. Dies ist das gleiche XML-Dokument, das auch von „GetCustomers“ und „GetSingleCustomers“ zurückgegeben wird.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

string **GetCustomerSelection***(string* ServerSession, *integer* SelectionID, *string* DataFields, *integer* MaxCount)

Gibt ein XML-Dokument mit den Kunden in der angegebenen Auswahl zurück.

SelectionID ist die ID der Auswahl.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

MaxCount ist die Maximalzahl der zurückgegebenen Kunden. MaxCount muss auf -1 eingestellt werden, um alle Kunden zurückzugeben, die die anderen angegebenen Kriterien erfüllen.

integer **GetCustomerSelectionCount***(string* ServerSession, *integer* SelectionID)

Gibt die Anzahl Kunden in der angegebenen Auswahl zurück.

SelectionID ist die ID der Auswahl.

string **GetCustomerSelectionSorted***(string* ServerSession, *integer* SelectionID, *string* DataFields, *string* SortFields, *integer* MaxCount)

Wie *GetCustomerSelection(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

string **GetCustomersSorted***(string* ServerSession, *integer* CustDomainID, *string* FilterExpression, *string* ContextExpression*, string* DataFields, *string* SortOrder, *integer* MaxCount)

Wie *GetCustomers(...)*, es kann jedoch die gewünschte Sortierung angegeben werden.

SortFields ist eine semikolongetrennte Liste von Kundendomänenfeldern, die zur Sortierung der zurückgegebenen Kunden verwendet werden sollen.

Hinweis: Sie können einem Sortierfeld das Präfix „~“ voranstellen, um eine absteigende Sortierung anzugeben.

string **GetExpressionFunctions**(*string* ServerSession)

Gibt ein XML-Dokument mit einer kategorisierten Beschreibung aller verfügbaren Funktionen zurück, die in Ausdrücken verwendet werden können. Dieses Dokument wird automatisch von Dialogue Server erstellt.

Hinweis: In der aktuellen Version ist ein Standardsatz von Ausdrucksfunktionen verfügbar. In zukünftigen Versionen wird die Möglichkeit implementiert, benutzerdefinierte Funktionen zu verwenden.

string **GetLookupDatarow**(*string* ServerSession, *integer* CustDomainID, *string* LookupSourceName, *string* KeyFieldname, *variant* KeyValue)

Gibt ein XML-Dokument mit Suchdaten zurück. Diese Methode gibt nur eine Zeile zurück, während *GetLookupDataset* alle Zeilen der Suchquelle zurückgibt.

CustDomainID ist die ID der Kundendomäne, für die die Suchquelle definiert ist.

LookupSourceName ist der Name der Suchquelle. *Schlüssel*

KeyFieldname ist der Name des Feldes in der Suchquelle, das zur Suche der zurückzugebenden Zeile verwendet wird.

KeyValue ist der Wert des mit *KeyFieldname* angegebenen Feldes in der gefundenen und zurückgegebenen Zeile.

string **GetLookupDataset**(*string* ServerSession, *integer* CustDomainID, *string* LookupSourceName)

Gibt ein XML-Dokument mit Suchdaten zurück. Diese Funktion gibt alle Zeilen in der Suchquelle zurück, während *GetLookupDatarow* nur eine Zeile zurückgibt.

CustDomainID ist die ID der Kundendomäne, für die die Suchquelle definiert ist.

LookupSourceName ist der Name der Suchquelle.

string **GetSelections**(*string* ServerSession)

Gibt ein XML-Dokument mit allen definierten Kundenauswahlen für alle Kundendomänen zurück. Diese Methode wird nicht mehr unterstützt. Verwenden Sie stattdessen die **GetSelections**-Methode der **Selection-API**.

string **GetSingleCustomer**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID*, string* DataFields)

Gibt ein XML-Dokument mit den Daten eines einzelnen Kunden zurück, der den angegebenen Kriterien entspricht.

CustDomainID ist die ID der Kundendomäne, *CustomerID* ist die ID des gewünschten Kunden.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

void **PostEvent**(*string* ServerSession, *string* EventTypeName*, string* Description*, integer* CustDomainID, *string* CustomerID*, string* Context)

Postet ein Systemereignis. Systemereignisse werden in Dialogue Admin definiert. Dialogue Server kann so eingerichtet werden, dass auf unterschiedliche Systemereignisse mit unterschiedlichen Aktivitäten reagiert wird. Beispielsweise kann ein Vorgang in einem oder mehreren Dialogfeldern ausgeführt werden.

EventTypeName ist die technische Bezeichnung des in Dialogue Admin definierten Ereignistyps.

Description ist die optionale Beschreibung eines Ereignisses.

CustDomainID ist die ID der Kundendomäne; *CustomerID* ist die ID des mit dem Ereignis verbundenen Kunden.

Context ist der Kontextwert des Kunden. Wenn kein Kontext benötigt wird, legen Sie *Context* auf die Zeichenfolge „0“ fest.

void **RemoveCategory***(string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* CategoryName)

Entfernt einen Kunden aus einer Kategorie, womit die Zugehörigkeit zu dieser Kategorie gelöscht wird.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

void **RemoveCategoryValue***(string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* CategoryName, *string* Value)

Entfernt einen Kundenwert in einer Kategorie. Die Kategorie muss den Typ *Kategorie mit Werten* haben.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Value den zu entfernenden String-Wert, der in der Kategorie Setup in Dialogue Admin definiert wird.

void **SetCategory***(string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* CategoryName)

Fügt einen Kunden zu einer Kategorie hinzu. Es wird für ihn also die Zugehörigkeit zu einer Kategorie erstellt.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

void **SetCategoryScore***(string* ServerSession, *integer* CustDomainID, *string* CustomerID*, string* CategoryName, *double* Score)

Fügt einen Kunden zu einer Kategorie mit Bewertungswert hinzu. Es wird also die Zugehörigkeit zu einer Kategorie erstellt oder der Bewertungswert einer bestehenden Zugehörigkeit aktualisiert.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Score ist der Bewertungswert zwischen den Minimal- und Maximalwerten, die in der Kategorieeinrichtung in Dialogue Admin definiert sind.

void **SetCategoryValue***(string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* CategoryName, *string Value*)

Fügt einen Kunden zu einer Kategorie mit Werten hinzu. Es wird also für ihn eine Zugehörigkeit zu einer Kategorie erstellt oder ein Wert zu einer bestehenden Zugehörigkeit hinzugefügt.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

CategoryName ist der technische Name der in Dialogue Admin definierten Kategorie.

Value bezeichnet den einzurichtenden String-Wert, der in der Kategorie Setup in Dialogue Admin definiert wird.

string **UpdateCustomerPassword**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *bool* AutoGenerate, *string* NewPassword)

Aktualisiert das Kennwort eines Kunden. Das neue Kennwort wird in der Datenbanktabelle CUST_LOGIN gespeichert und von dieser Methode zurückgegeben.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des Kunden.

Wenn *AutoGenerate* auf „true“ gesetzt ist, wird das neue Kennwort automatisch erstellt. Anderenfalls wird der Wert des Parameters *NewPassword* als neues Kennwort verwendet.

Hinweis: Es ist nur dann möglich, Kennwörter einzustellen, wenn in Dialogue Server die automatische Erstellung von Kennwörtern aktiviert wurde.

string **UpdateSingleCustomer**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID*, string* DataFields*, string* CustomerDataXML)

Sendet dem Dialogue Server Updates der Kundendaten zum Abspeichern der Änderungen. Mit dieser Methode kann ein neuer Kunde eingefügt oder ein bestehender Kunde gelöscht oder aktualisiert werden. Wenn ein neuer Kunde eingefügt wird, wird die Kunden-ID dieses Kunden zurückgegeben.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des zu aktualisierenden, zu löschenden oder einzugebenden Kunden.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die in „CustomerDataXML“ enthalten sind.

CustomerDataXML ist ein XML-Dokument mit Kundendaten. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams. Wenn eine von *GetCustomers* oder *GetSingleCustomer* erhaltene XML in einen ADO-Datensatz geladen wird und dieser Datensatz modifiziert wird, kann ADO.NET automatisch das Diffgram erstellen, welches gesendet wird an: *UpdateSingleCustomer* .

Hinweis: *DataFields* enthält theoretisch redundante Informationen. Allerdings ist diese in der aktuellen Version erforderlich, um die Kommunikation von Änderungen zwischen einem Client und dem Dialogue Server zu erleichtern.

string **UpdateSingleCustomerSimple**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* CustomerDataXML)

Sendet dem Dialogue Server Updates der Kundendaten zum Abspeichern der Änderungen. Diese Methode kann einen neuen Kunden hinzufügen oder einen existierenden Kunden aktualisieren. Wenn ein neuer Kunde eingefügt wird, wird die Kunden-ID dieses Kunden zurückgegeben.

CustDomainID entspricht der ID der Kundendomäne und *CustomerID* entspricht der ID des zu aktualisierenden, zu löschenden oder einzugebenden Kunden.

CustomerDataXML ist ein XML-Dokument mit den veränderten oder neuen Kundendaten. Die Struktur des XML-Dokuments entspricht den XML-Daten, die von *GetSingleCustomer* zurückgegeben werden (siehe oben).

Hinweis: „UpdateSingleCustomerSimple“ unterstützt zum jetzigen Zeitpunkt nicht das Löschen von Kunden oder Dateneinträgen von Kunden.

Beispiele

Im folgenden Beispiel werden Kundeninformationen von Dialogue Server abgerufen. Das Beispiel verwendet JScript.

```
   ..............
   //Connect to the DialogServer
   SystemAPI = new ActiveXObject("MHDialogServer.MHSystemAPI");

   //Log in
   var InstanceName = "MHProduction";
   var SessionKey = SystemAPI.Login(InstanceName, "admin", "ringo1", "MH
Test");
   var ServerSession = SessionKey + "@" + InstanceName;

   //Now get some customers
   CustomerAPI = new ActiveXObject("MHDialogServer.MHCustomerAPI");
  var CustomerXML = CustomerAPI.GetCustomers(ServerSession, 1000, "Address.Ci-
ty=\"Oslo\"", "", "", -1);

   //Log out (we got what we wanted)
```

```
   SystemAPI.Logout(ServerSession);
   ..............
```

Das nächste Beispiel evaluiert einen Ausdruck, um zu ermitteln, ob eine Popup-Nachricht auf einer Webseite angezeigt werden soll oder nicht.

```
   ..............
   CustomerAPI = new ActiveXObject("MHDialogServer.MHCustomerAPI");

   //Check if the customer fulfills our criteria for displaying the pop-up
message
   var DisplayPopup = CustomerAPI.EvaluateExpressionBool(ServerSession, 1000,
 CustomerID,
      "Age>25 and Address.Town=\"London\"");
   ..............
```

Generic-API

Übersicht

Die Generic-API stellt eine Reihe von Methoden zum Zugriff auf in Dialogue Admin definierte generische Objekte zur Verfügung. Dies umfasst sowohl Datensätze und in der SQL-Repository gespeicherte SQL-Anweisungen als auch die Ausführung von generischen Plug-Ins.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHGenericAPI* implementiert die Generic-API über die Schnittstelle *IMHGenricAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

variant **ExecutePlugin**(*string* ServerSession, *string* PluginName, *array* Params)

variant **ExecutePlugin_JS**(*string* ServerSession, *string* PluginName, *array* Params)

Führt ein Dialogue Server-Plug-In vom Typ *Generisches Plug-In* aus, das die Schnittstelle **IMHGenericPlugin** implementiert. *PluginName* ist der in Dialogue Admin *definierte Name des Plug-Ins. Params* ist ein Array, das Parameterwerte enthält.

Unter dem Rückgabewert versteht man den vom Plug-In zurückgegebenen Wert.

Hinweis: *ExecutePlugin_JS* ist eine alternative, für die Verwendung mit JScript und den in JScript verwendeten Arraytypen geeignete Implementierung.

integer **ExecuteSQL**(*string* ServerSession, *string* SQLName, *array* Params)

integer **ExecuteSQL_JS**(*string* ServerSession, *string* SQLName, *array* Params)

Führt eine im Dialogue Admin SQL Repository gespeicherte SQL-Anweisung aus. *SQLName* ist der technische Name der Anweisung. *Params* ist ein Array, das Parameterwerte enthält.

„SQLExecute“ gibt die Anzahl der betroffenen Datensätze zurück.

Hinweis: *ExecuteSQL_JS* ist eine alternative, für die Verwendung mit JScript und den in JScript verwendeten Arraytypen geeignete Implementierung.

void **ExecuteSQLScript**(*string* ServerSession, *string* SQLName, *array of MHGenericParam* Params)

Führt ein im Dialogue Admin SQL Repository gespeichertes SQL-Skript (mehrere SQL-Anweisungen) aus. *SQLName* ist der technische Name der Anweisung.

Params ist ein Array, das Parameterwerte enthält. Das Array enthält Paare jeweils aus Parametername und -wert, die in COM (Microsoft IDL) definiert sind als:

```
typedef struct tagMHGenericParam { BSTR Name; VARIANT Value; } MHGenericPa-
ram;
```

Die Definition von *MHGenericParam* kann aus einem Datensatz, einer Struktur oder Klasse bestehen, abhängig von der Aufrufsprache. Zum Beispiel wird „MHGenericParam“ in C# bei Verwendung einer *Webreferenz* zur Dialogue Server-API automatisch als Klasse deklariert.

Hinweis: Diese Methode ermöglicht es Clients, eine Folge von SQL-Anweisungen in einer Transaktion auszuführen.

void **ExecuteStoredProc**(*string* ServerSession, *string* StoredProcName, *array* Params, *string* ConnectionName)

void **ExecuteStoredProc_JS**(*string* ServerSession, *string* StoredProcName, *array* Params, *string* ConnectionName)

Führt eine gespeicherte Prozedur mit der angegebenen Datenbankverbindung durch.

„StoredProcName“ ist der in der Datenbank definierte Name der gespeicherten Prozedur, während *Params* ein Array mit Parameterwerten ist.

ConnectionName ist der technische Name der in Dialogue Admin definierten sekundären Datenbankverbindung. Ein leerer ConnectionName bedeutet, dass die Standard Datenbankverbindung verwendet werden sollte.

Hinweis: *ExecuteStoredProc_JS* ist eine alternative, für die Verwendung mit JScript und den in JScript verwendeten Arraytypen geeignete Implementierung.

string **GetDataset**(*string* ServerSession, *string* SQLName, *array* Params, *integer* MaxRows)

string **GetDataset_JS**(*string* ServerSession, *string* SQLName, *array* Params, *integer* MaxRows)

Gibt einen in Dialogue Admin als XML definierten Datensatz zurück.

SQLName ist der Name des den Datensatz definierenden Elements im SQL-Repository.

Params ist ein Array mit Parameterwerten (Datenbank-Hostvariablen).

MaxRows legt die maximale Anzahl der abzurufenden Zeilen fest. Wenn *MaxRows* auf -1 festgelegt wird, werden alle Zeilen zurückgegeben, während bei einem entsprechenden Wert von 0 nur das XML-Schema zurückgegeben wird.

Hinweis: *GetDataset_JS* ist eine alternative, für die Verwendung mit JScript und den in JScript verwendeten Arraytypen geeignete Implementierung.

int64 **GetSequenceID**(*string* ServerSession, *string* SequenceName)

Gibt den nächsten Wert der angegebenen Sequenz zurück. *Sequences* ist ein von einem Dialogue Server zur Verfügung gestellter Mechanismus zur Aufrechterhaltung eindeutiger Kennungen. Durch den Dialogue Server wird die Einzigartigkeit der zurückgegebenen Zahl in unterschiedlichen Aufrufen der „GetSequenceID“ garantiert.

SequenceName ist der Name der Sequenz. Wenn eine Sequenz festgelegt wird, die nicht existiert, wird automatisch eine neue Sequenz erstellt.

Hinweis: Sequenzen sind interne Mechanismen in Dialogue Server und sollten nicht mit Datenbanksequenzen verwechselt werden. Zum Beispiel verfügt Oracle über einen eigenen, als *Sequenzen* bezeichneten Mechanismus.

variant **UpdateDataset**(*string* ServerSession, *string* SQLName, *array* Params, *string* XMLDiffGram)

variant **UpdateDataset_JS**(*string* ServerSession, *string* SQLName, *array* Params, *string* XMLDiffGram)

Aktualisiert einen Datensatz, der vorher mit der *GetDataset*-Methode abgerufen wurde.

UpdateDataset gibt die letzte von der Datenbank generierte ID für die im Datensatz enthaltene Tabelle auf höchster Ebene zurück. Typischerweise handelt es sich dabei um den Wert einer automatisch inkrementierten Spalte oder einer Datenbanksequenz. Wenn die entsprechenden Spalten nicht in Dialogue Admin definiert wurden, ist der Rückgabewert *null.*

SQLName ist der Name des den Datensatz definierenden Elements im SQL-Repository.

Params ist ein Array von Parameterwerten. Diese Parameterwerte müssen den Parameterwerten entsprechen, die festgelegt wurden, als der Originaldatensatz mit *GetDataset* abgerufen wurde.

XMLDiffGram ist ein XML-Dokument, das neue, aktualisierte oder gelöschte Datensätze enthält. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams.

Hinweis: *UpdateDataset_JS* ist eine alternative, für die Verwendung mit JScript und den in JScript verwendeten Arraytypen geeignete Implementierung.

Beispiele

Im folgenden Beispiel wird ein generisches Plug-In ausgeführt. Das Beispiel verwendet C#, um den SOAP-Dienst des Generic-API aufzurufen.

```
   ..............
   //Login
  SystemAPI.SystemAPIService srvSystemAPI = new SystemAPI.SystemAPIService();

   string ServerSession = srvSystemAPI.Login("default", "a", "a", "Test Sys-
tem");
   ServerSession = ServerSession + "@default";
```

```
   //Create generic API object to execute a plug-in
   GenericAPI.GenericAPIService srvGenericAPI = new GenericAPI.GenericAPISer-
vice();
   //Create an array of input parameters to the plug-in
   Object[] Params = new Object[3] { "Herman Villsvinsen", 1004, new DateTi-
me(1978, 04, 13) };
   Object ReturnValue = srvGenericAPI.ExecutePlugin(ServerSession, "Execute-
Test", Params);
   //Show return value in a windows message box
   MessageBox.Show( "Return value: " + ReturnValue.ToString() );

   //Logout
   srvSystemAPI.Logout(ServerSession);
   ..............
```

Message-API

Übersicht

Die Mail-API bietet eine Reihe von Methoden zum Zugriff auf Daten im Zusammenhang mit Kundennachrichten.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHMessageAPI* implementiert die Message-API durch die Schnittstelle *IMHMessageAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

string **BuildEmailTestReportUrl(** *string* ServerSession, *integer* TestID **)**

BuildEmailTestReportUrl gibt die URL einer Webseite mit dem Bericht über einen E-Mail-Test zurück.

TestID ist die eindeutige Kennung des Tests. Neue Tests werden mit dem Aufruf der Methode *CreateEmailTest(...)* erstellt, welche eine eindeutige ID des neu erstellten Tests zurückgibt. Die IDs bereits vorhandener Tests können mit der Methode *GetEmailTestLogs(...)* abgerufen werden.

integer **CreateEmailTest**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *variant* ContentBinary, *string* ControlParams, *variant* MasterTemplateID, *variant* BaseMessageID, *variant* MessageBundleID, *out string* ReportUrl)

CreateEmailTest ruft einen externen Dienst auf, um eine E-Mail-Nachricht mit verschiedenen E-Mail-Clients und Spamfiltern zu testen. Der zurückgegebene Wert ist eine eindeutige Kennung des Tests, und die URL verweist auf eine Webseite mit dem Testbericht.

MessageTypeID ist die eindeutige ID des in Dialogue Admin konfigurierten Nachrichtentyps.

ContentText ist der Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

MasterTemplateID: Wenn die getestete Nachricht eine Mastervorlage ist, ist *MasterTemplateID* ihre ID, ansonsten ist der Wert *null.*

BaseMessageID: Wenn die getestete Nachricht eine normale Vorlage (Basisnachricht) ist, ist *BaseMessageID* ihre ID, ansonsten ist der Wert *null.*

MessageBundleID: Wenn die getestete Nachricht eine Nachricht in einem Nachrichtenpaket ist, ist *MessageBundleID* ihre ID, ansonsten ist der Wert *null.*

ReportUrl ist ein Ausgabeparameter der eine URL enthält, die auf die Webseite mit dem Testbericht verweist.

Hinweis: Normalerweise eine und mindestens eine der o.g. IDs (MasterTemplateID, BaseMessageID oder MessageBundleID) sollten angegeben werden.

string **CreateMergeFile**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* DataFields, *integer* MaxCount*, bool* UseCharDelimiter*, bool* UseQuote*, string* CharDelimiter*, string* QuoteChar *)*

Erstellt eine einfache Datei mit Kundeninformationen. Die zurückgegebene Zeichenfolge kann von der aufrufenden Anwendung in einer Datei gespeichert werden. *CreateMergeFile* kann mehrere Kunden in einer Seriendruckdatei verarbeiten, während *CreateSingleMergeFile* nur einen einzelnen Kunden behandelt.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

FilterExpression ist ein Ausdruck zur Bestimmung der Kunden, die in die Seriendruckdatei aufgenommen werden sollen. *ContextExpression* ist ein optionaler Ausdruck zum Festlegen eines Kontextwerts.

MaxCount ist die Maximalzahl der Kunden, die in die Seriendruckdatei aufgenommen werden können. Setzen Sie „MaxCount“ auf -1, um diese Beschränkung der Kundenzahl aufzuheben.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern, die in der Datei enthalten sein sollen. Felder aus Datengruppen, die mehrere Zeilen pro Kunde zurückgeben, werden nur unterstützt, wenn in Dialogue Admin ein Ausdruck zum Erhalten einer Standardzeile angegeben wurde.

UseCharDelimiter gibt an, ob die Spalten in der Datei mit einem Zeichen separiert werden sollen. Anderenfalls wird die Trennung mit Tabulatoren verwendet. *CharDelimiter* gibt das Trennzeichen an.

UseQuote gibt an, ob Feldwerte in Anführungszeichen gesetzt werden sollen. *QuoteChar* ist das einzelne Zeichen, das als Anführungszeichen verwendet werden soll.

Hinweis: Das Ergebnis ist für Seriendruckvorgänge geeignet. Diese Methode kann auch verwendet werden, um Kundendaten in Containerdateien zu exportieren.

string **CreateSingleMergeFile**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* DataFields, *bool* UseCharDelimiter*, bool* UseQuote*, string* CharDelimiter*, string* QuoteChar *)*

Erstellt eine einfache Datei mit Kundeninformationen. Die zurückgegebene Zeichenfolge kann von der aufrufenden Anwendung in einer Datei gespeichert werden. *CreateMergeFile* kann mehrere Kunden in einer Seriendruckdatei verarbeiten, während *CreateSingleMergeFile* nur einen einzelnen Kunden behandelt.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

CustomerID ist die ID des Kunden, der in die Seriendruckdatei aufgenommen wird.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern, die in der Datei enthalten sein sollen. Felder aus Datengruppen, die mehrere Zeilen pro Kunde zurückgeben, werden nur unterstützt, wenn in Dialogue Admin ein Ausdruck zum Erhalten einer Standardzeile angegeben wurde.

UseCharDelimiter gibt an, ob die Spalten in der Datei mit einem Zeichen separiert werden sollen. Anderenfalls wird die Trennung mit Tabulatoren verwendet. *CharDelimiter* gibt das Trennzeichen an.

UseQuote gibt an, ob Feldwerte in Anführungszeichen gesetzt werden sollen. *QuoteChar* ist das einzelne Zeichen, das als Anführungszeichen verwendet werden soll.

Hinweis: Das Ergebnis ist für Seriendruckvorgänge geeignet. Diese Methode wird von Customer View zur Erstellung von Datendateien bei der Bereitstellung von Word-Serienbriefen verwendet.

string **CreateSingleWordMergeFile**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* BaseMessageID, *string* DataFields *)*

Erstellt eine einfache Seriendruck-Datei für MS Word Mail Merge. Die Datei wird als Zeichenfolge zurückgegeben und enthält Informationen über einen einzelnen Kunden zur Verwendung mit der angegebenen Nachrichtenvorlage.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

CustomerID ist die ID des Kunden, der in die Seriendruckdatei aufgenommen wird.

BaseMessageID ist die ID der in Visual Dialogue erstellten Nachrichtenvorlage.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern, die in der Datei enthalten sein sollen. Felder aus Datengruppen, die mehrere Zeilen pro Kunde zurückgeben, werden nur unterstützt, wenn in Dialogue Admin ein Ausdruck zum Erhalten einer Standardzeile angegeben wurde.

UseCharDelimiter gibt an, ob die Spalten in der Datei mit einem Zeichen separiert werden sollen. Anderenfalls wird die Trennung mit Tabulatoren verwendet. *CharDelimiter* gibt das Trennzeichen an.

UseQuote gibt an, ob Feldwerte in Anführungszeichen gesetzt werden sollen. *QuoteChar* ist das einzelne Zeichen, das als Anführungszeichen verwendet werden soll.

Hinweis: Diese Methode ist zur Unterstützung der Word Mail Merge-Funktion in Customer View implementiert worden.

void **DeleteCustomerMessage**(*string* ServerSession, *int64* CustomerMessageID)

Löscht eine Nachricht, die einem bestimmten Kunden zugeordnet ist. Eine Nachricht kann mit mehreren Kunden in Beziehung stehen. Wenn keine derartigen Beziehungen mehr bestehen, wird die entspre-

chende Nachricht gelöscht. Wenn der Kunde über eine Aktivität verfügt, die beim Verschicken der Nachricht erstellt wurde, wird diese Aktivität gelöscht.

CustomerMessageID ist die eindeutige ID der Beziehung zwischen einer Nachricht und einem Kunden (Feld CM_ID in Datenbanktabelle CUSTOMER_MESSAGE).

void **DeleteEmailTestLog(** *string* ServerSession, *integer* EmailTestLogID **)**

DeleteEmailTestLog löscht einen bestehenden E-Mail-Test für die Anmeldung in der Datenbank.

EmailTestLogID ist die eindeutige Kennung des in der Datenbanktabelle EMAIL_TEST_LOG protokollierten Testberichts. Bestehende E-Mail-Tests können mithilfe des Aufrufs *GetEmailTestLogs(..)* abgerufen werden.

void **DeleteMessage**(*string* ServerSession, *int64* MessageID)

Löscht eine Nachricht. Eine Nachricht kann mit einem oder mehreren Kunden verknüpft werden. Alle diese Verknüpfungen werden gelöscht. Außerdem werden alle Aktivitäten gelöscht, die erstellt wurden, als die Nachricht abgeschickt wurde.

MessageID ist die eindeutige ID der Nachricht (Feld ML_ID in der Datenbanktabelle MESSAGE_LOG).

void **DeleteTemplate**(*string* ServerSession, *integer* BaseMessageID, *bool* DeleteRelatedMessages)

Löscht eine Nachrichtenvorlage.

BaseMessageID ist die ID der Vorlage.

DeleteRelatedMessages legt fest, ob aus dieser Vorlage erstellte Nachrichten zusammen mit der Vorlage gelöscht werden sollen. Wenn *false*, werden auf dieser Vorlage basierende Nachrichten nicht gelöscht.

*string***GetControlParemeterDefinitions**(*string* ServerSession, *integer* DocumentTypeID)

Gibt die Steuerparameterdefinitionen für einen bestimmten Dokumententyp als XML zurück.

DocumentTypeID ist die ID des Dokumententyps, für den Informationen abgerufen werden.

string **GetCustomSqlFieldInfo**(*string* ServerSession, *variant SqlConnectionID,* *string* SqlStatement)

Gibt als XML Informationen über die Felder im Resultset eines benutzerdefinierten SQL-Befehls zurück, der in einer Vorlage verwendet wird.

SqlConnectionID enthält die ID der Datenbankverbindung, die zur Ausführung des benutzerdefinierten SQL-Befehls verwendet wird. Der Wert *null* steht für die Standardverbindung.

SQLStatement ist der benutzerdefinierte SQL-Befehl.

int64 **GetEmailSpamRate**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *variant* ContentBinary, *string* ControlParams, *bool* IncludeHtmlReport, *out string* HtmlReport)

GetEmailSpamRate ruft einen externen Dienst zur Berechnung des Werts einer Spambewertung für eine E-Mail-Nachricht auf. Der zurückgegebene Wert entspricht dem Wert der Spambewertung. Optional wird dadurch ebenfalls ein Spambewertungsbericht im HTML-Format erstellt.

MessageTypeID ist die eindeutige ID des in Dialogue Admin konfigurierten Nachrichtentyps.

ContentText ist der Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

IncludeHtmlReport legt fest, ob ein Spambewertungsbericht im HTML-Format erstellt und zurückgegeben werden soll.

HtmlReport ist ein Ausgabeparameter, der den generierten Spambewertungsbericht im HTML-Format enthält.

string **GetEmailTestLogs(** *string* ServerSession, *variant* MasterTemplateID, *variant* BaseMessageID, *variant* MessageBundleID **)**

GetEmailTestLogs gibt sämtliche protokollierten Tests einer festgelegten E-Mail-Vorlage oder E-Mail-Nachricht als XML zurück.

MasterTemplateID: Wenn die abzurufenden E-Mail-Tests mit einer Mastervorlage in Beziehung stehen, enthält *MasterTemplateID* die ID der Mastervorlage. Anderenfalls ist der Wert *null.*

BaseMessageID: Wenn die abzurufenden E-Mail-Tests mit einer normalen Vorlage (Basisnachricht) in Beziehung stehen, enthält *BaseMessageID* die ID der Vorlage. Anderenfalls ist der Wert *null.*

MessageBundleID: Wenn die abzurufenden E-Mail-Tests mit einer Nachricht in einem Nachrichtenbündel in Beziehung stehen, enthält *MessageBundleID* die ID des Nachrichtenbündels. Anderenfalls ist der Wert *null.*

Hinweis: Normalerweise sollte mindestens eine der oben beschriebenen IDs festgelegt werden.

string **GetFuzzyMessages**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle definierten Fuzzy-Nachrichten enthält.

string **GetMasterTemplate**(*string* ServerSession, *integer* MasterTemplateID)

Gibt eine bestimmte Mastervorlage als XML zurück.

MasterTemplateID ist die eindeutige Kennung der Mastervorlage.

string **GetMasterTemplateDetails**(*string* ServerSession, *integer* MasterTemplateID)

Diese Methode wird nicht mehr unterstützt. Verwenden Sie stattdessen die *GetMasterTemplate*-Methode.

string **GetMasterTemplates**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle definierten Mastervorlagen enthält. Mastervorlagen werden in Visual Dialogue geschrieben oder entworfen. Das XML-Dokument gruppiert die Mastervorlagen nach Kundendomänen.

string **GetMessageAttachment**(*string* ServerSession, *int64* MessageID, *integer* AttachmentIndex)

Gibt ein XML-Dokument zurück, das den Dateinamen und den Inhalt eines Anhangs einer Nachricht enthält. Diese Methode kann zum Herunterladen von Anhängen in einer E-Mail-Nachricht verwendet werden. Der Inhalt der Anhangsdatei wird in der XML mittels *Base64Binary*-Codierung verschlüsselt.

MessageID ist die ID der Nachricht.

AttachmentIndex ist eine Ganzzahl, die die Position des Anhangs im mit *Attachments* bezeichneten Nachrichtensteuerparameter enthält. Dieser Steuerparameter sollte eine durch Semikola getrennte Liste der an die Nachricht angehängten Dateinamen enthalten.

Hinweis: In der Standardinstallation können ausschließlich E-Mail-Nachrichten über Anhänge verfügen.

string **GetMessageAttachmentSchema**(*string* ServerSession)

Gibt das Schema der XML zurück, die zur Festlegung von Anhängen in Aufrufen der Methoden *PostSingleMessageEx(..)* und *PostSingleDialogMessageEx(..)* verwendet wird.

string **GetMessageDetails**(*string* ServerSession, *int64* MessageID, *boolean* IncludeContent)

Gibt ein XML-Dokument zurück, das Details zu einer bestimmten Nachricht enthält.

MessageID ist die ID der Nachricht.

IncludeContent legt fest, ob der Nachrichteninhalt eingeschlossen ist. Wenn auf *true* gesetzt, ist der Inhalt (als Text oder binär) in der zurückgegeben XML eingeschlossen. Binärdaten werden mittels *Base64Binary*-Codierung verschlüsselt.

string **GetMessageDetailsIdent**(*string* ServerSession, *string* MessageIdentifier, *boolean* IncludeContent)

Gibt ein XML-Dokument zurück, das Details zu einer bestimmten Nachricht enthält. Das XML-Dokument ist das gleiche wie das durch *GetMessageDetails(...)* zurückgegebene Dokument.

MessageIdentifier ist eine eindeutige Zeichenfolge, welche die Nachricht identifiziert. Diese Kennung wird bei der Erstellung einer neuen Nachricht in Dialogue Server erstellt und im Datenbankfeld ML_MESSAGE_IDENTIFIER gespeichert.

IncludeContent legt fest, ob der Nachrichteninhalt eingeschlossen ist. Wenn auf *true* gesetzt, ist der Inhalt (als Text oder binär) in der zurückgegeben XML eingeschlossen. Binärdaten werden mittels *Base64Binary*-Codierung verschlüsselt.

string **GetMessagesLockedByUser**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das eine Liste von Nachrichten enthält, die durch den aktuellen Benutzer zur Bearbeitung gesperrt wurden.

string **GetMessageTypes**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das festgelegte Nachrichtentypen enthält. Nachrichtentypen werden in Dialogue Admin konfiguriert.

string **GetTemplate**(*string* ServerSession, *integer* BaseMessageID)

Gibt eine bestimmte Nachrichtenvorlage als XML zurück.

BaseMessageID ist die eindeutige Kennung der Nachrichtenvorlage.

string **GetTemplateDetails**(*string* ServerSession, *integer* BaseMessageID)

Diese Methode wird nicht mehr unterstützt. Verwenden Sie stattdessen die *GetTemplate*-Methode.

string **GetTemplates**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle definierten Nachrichtenvorlagen enthält. Vorlagen werden in Visual Dialogue geschrieben oder entworfen. Das XML-Dokument gruppiert die Nachrichtenvorlagen nach Kundendomänen.

string **GetTemplateSchema**(*string* ServerSession)

Gibt das für Nachrichtenvorlagen verwendete XML-Schema zurück. Dieses Schema wird als Parameter bzw. Rückgabewert von den Methoden *GetTemplate(...)* und *SaveTemplate(...)* verwendet.

void **LockMessage**(*string* ServerSession, *int64* MessageID, *string* ClientLocalPath, *string* ClientMachine)

Sperrt eine bestimmte Nachricht für die Bearbeitung.

MessageID ist die eindeutige ID der zu sperrenden Nachricht.

ClientLocalPath ist ein optionaler Parameter, der den lokalen Pfad festlegt, auf dem während der Bearbeitung eine lokale Kopie der Nachricht bereitgehalten wird. *ClientMachine* ist ein optionaler Parameter, der die Bezeichnung des Computers, auf dem die Nachricht bearbeitet wird, bereithält. Sowohl *ClientLocalPath* als auch *ClientMachine* werden vom System gespeichert, und die entsprechenden Werte können später durch Aufrufe von *GetMessageDetails* abgerufen werden. Diese beiden Werte haben nur in Client-Anwendungen eine Bedeutung, in denen die Sperrung, Aktualisierung und Entsperrung von Nachrichten während der Bearbeitung gesteuert wird.

int64 **PostSingleDialogMessage(** *string* ServerSession, *integer* CustDomainID, *int64* ParticipantID, *string* MessageTypeName, *integer* BaseMessageID, *string* ContentText, *variant* ContentBinary, *string* ControlParams, *string* MessageName, *string* ActivityTypeName, *string* ActivityDesc, *bool* UseOutbox,*bool* MarkAsSent, *string* Direction, *datetime* Timestamp, *integer* Priority)

CustDomainID ist die ID der Kundendomäne. *ParticipantID* ist die ID des Teilnehmers, mit dem die Nachricht in Beziehung steht. Die Nachricht steht außerdem mit dem Kunden in Beziehung, auf den vom Teilnehmer gezeigt wird, allerdings werden *CustomerID* und *Context* intern von Dialogue Server aufgelöst.

MessageTypeName ist der in Dialogue Admin entsprechend konfigurierte technische Name des Nachrichtentyps.

BaseMessageID ist eine optionale Referenz auf eine Nachrichtenvorlage (definiert in Visual Dialogue). Dadurch zeigt eine Nachricht auf die Vorlage, auf deren Grundlage sie erstellt wurde. Legen Sie *BaseMessageID* auf -1 fest, wenn dies nicht zutrifft.

ContentText ist der Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

MessageName ist der Name der Nachricht.

ActivityTypeName ist der technische Name des zu versendenden Aktivitätstyps beim Speichern der Nachricht. Wenn kein Wert festgelegt wurde, wird keine Aktivität versendet. *ActivityDesc* ist die Beschreibung der Aktivität und sollte über einen Wert verfügen, wenn ein Aktivitätstyp festgelegt wurde.

Wenn „true“, wird durch *UseOutbox* festgelegt, dass die Nachricht im Postausgang anstatt in der Sendewarteschlange (für elektronische Nachrichten) bzw. in den gesendeten Elementen (für nicht-elektronische Nachrichten) gespeichert werden soll. Im Postausgang gespeicherte Nachrichten können später mit dem Message Manager in Visual Dialogue in die Sendewarteschlange (für elektronische Nachrichten) bzw. zu den gesendeten Elementen (für nicht-elektronische Nachrichten) verschoben werden.

Wenn „true“, wird durch *MarkAsSent* festgelegt, dass die Nachricht zusammen mit den gesendeten Elementen und nicht in der Sendewarteschlange bzw. im Postausgang gespeichert werden soll.

Direction beschreibt, ob die Nachricht vom Kunden oder von „uns“ bzw. „dem System“ erstellt wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN", vom Kunden.
- Direction = "OUT", von „uns“ bzw. „dem System“.

Timestamp ist der Zeitpunkt, zu dem die Nachricht gesendet wurde.

Priority steuert die Priorität der Nachricht beim elektronischen Versenden von Nachrichten, z. B. bei E-Mails oder SMS-Nachrichten. *Priority* kann die folgenden fünf möglichen Werte haben:

Wert	Wertname	Beschreibung
-2	Niedrig	Die niedrigste Priorität. Diese Nachrichten werden versendet, nachdem alle anderen Nachrichten in der Sendewarteschlange verarbeitet wurden.
-1	Medium low	Mittelniedrige Priorität.
0	Normal	Die standardmäßige Nachrichtenpriorität.
1	Medium high	Mittelhohe Priorität.
2	Hoch	Die höchste Priorität. Diese Nachrichten werden vor allen anderen Nachrichten in die Sendewarteschlange gesendet.

int64 **PostSingleDialogMessageEx(** *string* ServerSession, *integer* CustDomainID, *int64* ParticipantID, *string* MessageTypeName, *integer* BaseMessageID, *string* ContentText, *variant* ContentBinary, *string* ControlParams, *string* MessageName, *string* ActivityTypeName, *string* ActivityDesc, *bool* UseOutbox, *bool* MarkAsSent, *string* Direction, *datetime* Timestamp, *integer* Priority, *string* AttachmentsXML)

Diese Methode bewirkt das Gleiche wie *PostSingleDialogMessage(..)*, außer dass dadurch zusätzlich zur Nachricht eine Reihe von Anhangsdateien gespeichert wird. Der Steuerparameter *Attachments* wird automatisch aktualisiert, um die gespeicherten Anhänge widerzuspiegeln.

AttachmentsXML ist eine XML-Datei, die die zu speichernden Anhänge enthält. Zum Abrufen des Schemas dieser XML-Datei rufen Sie die Methode *GetMessageAttachmentSchema(..)* auf.

int64 **PostSingleMessage(** *string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *string* MessageTypeName, *integer* BaseMessageID*, string* ContentText, *variant* ContentBinary, *string* ControlParams, *string* MessageName, *string* ActivityTypeName, *string* ActivityDesc, *bool* UseOutbox, *bool* MarkAsSent, *string* Direction, *datetime* Timestamp, *integer* Priority)

CustDomainID ist die ID der Kundendomäne. *CustomerID* ist die ID des Kunden, der mit dieser Nachricht in Beziehung steht. *Context ist ein optionaler Kontextwert. Der Wert „0“* bedeutet, dass kein Kontext vorhanden ist.

MessageTypeName ist der in Dialogue Admin entsprechend konfigurierte technische Name des Nachrichtentyps.

BaseMessageID ist eine optionale Referenz auf eine Nachrichtenvorlage (definiert in Visual Dialogue). Dadurch zeigt eine Nachricht auf die Vorlage, auf deren Grundlage sie erstellt wurde. Legen Sie *BaseMessageID* auf -1 fest, wenn dies nicht zutrifft.

ContentText ist der Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

MessageName ist der Name der Nachricht.

ActivityTypeName ist der technische Name des zu versendenden Aktivitätstyps beim Speichern der Nachricht. Wenn kein Wert festgelegt wurde, wird keine Aktivität versendet. *ActivityDesc* ist die Beschreibung der Aktivität und sollte über einen Wert verfügen, wenn ein Aktivitätstyp festgelegt wurde.

Wenn „true“, wird durch *UseOutbox* festgelegt, dass die Nachricht im Postausgang anstatt in der Sendewarteschlange (für elektronische Nachrichten) bzw. in den gesendeten Elementen (für nicht-elektronische Nachrichten) gespeichert werden soll. Im Postausgang gespeicherte Nachrichten können später mit dem Message Manager in Visual Dialogue in die Sendewarteschlange (für elektronische Nachrichten) bzw. zu den gesendeten Elementen (für nicht-elektronische Nachrichten) verschoben werden.

Wenn „true“, wird durch *MarkAsSent* festgelegt, dass die Nachricht zusammen mit den gesendeten Elementen und nicht in der Sendewarteschlange bzw. im Postausgang gespeichert werden soll.

Direction beschreibt, ob die Nachricht vom Kunden oder von „uns“ bzw. „dem System“ erstellt wurde. „Direction“ kann die folgenden Werte haben:

- Direction = "IN", vom Kunden.
- Direction = "OUT", von „uns“ bzw. „dem System“.

Timestamp ist der Zeitpunkt, zu dem die Nachricht gesendet wurde.

Priority steuert die Priorität der Nachricht beim elektronischen Versenden von Nachrichten, z. B. bei E-Mails oder SMS-Nachrichten. Siehe *PostSingleDialogMessage(...)* oben für mögliche Werte von *Priority.*

int64 **PostSingleMessage(** *string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *string* MessageTypeName, *integer* BaseMessageID*, string* ContentText, *variant* ContentBinary, *string* ControlParams, *string* MessageName, *string* ActivityTypeName, *string* ActivityDesc, *bool* UseOutbox, *bool* MarkAsSent, *string* Direction, *datetime* Timestamp, *integer* Priority, *string* AttachmentsXML)

Diese Methode bewirkt das Gleiche wie *PostSingleMessage(..)*, außer dass dadurch zusätzlich zur Nachricht eine Reihe von Anhangsdateien gespeichert wird. Der Steuerparameter *Attachments* wird automatisch aktualisiert, um die gespeicherten Anhänge widerzuspiegeln.

AttachmentsXML ist eine XML-Datei, die die zu speichernden Anhänge enthält. Zum Abrufen des Schemas dieser XML-Datei rufen Sie die Methode *GetMessageAttachmentSchema(..)* auf.

integer **ProducePostMessages**(*string* ServerSession, *integer* CustDomainID, *string* FilterExpression, *string* ContextExpression, *integer* BaseMessageID, *integer* MaxCount*, string* MessageName, *string* ActivityDesc)

Erstellt und speichert eine Nachricht aus einer bestimmten Vorlage für den Kundensatz. Es wird die Anzahl der mit der/den erstellten Nachricht(en) verbundenen Kunden zurückgegeben.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

FilterExpression ist ein Ausdruck zur Bestimmung der einzubeziehenden Kunden. *ContextExpression* ist ein optionaler Ausdruck zum Festlegen eines Kontextwerts.

BaseMessageID ist die ID der zu verwendenden Vorlage.

MaxCount ist die maximale Anzahl an Kunden. Setzen Sie „MaxCount“ auf -1, um diese Beschränkung der Kundenzahl aufzuheben.

MessageName ist der Name der Nachricht.

ActivityDesc ist die Beschreibung der Aktivität und muss über einen Wert verfügen, wenn ein Aktivitätstyp zum Versenden für die verwendete Vorlage festgelegt wurde.

int64 **ProducePostSingleDialogMessage**(*string* ServerSession, *integer* CustDomainID, *int64* ParticipantID, *integer* BaseMessageID, *string* MessageName, *string* ActivityDesc)

Erstellt und speichert eine Nachricht aus einer bestimmten Vorlage für den festgelegten Dialogteilnehmer. Die ID der neuen Nachricht wird zurückgegeben.

CustDomainID ist die ID der Kundendomäne. *ParticipantID* ist die ID des Teilnehmers, mit dem die Nachricht in Beziehung steht. Die Nachricht steht außerdem mit dem Kunden in Beziehung, auf den vom Teilnehmer gezeigt wird, allerdings werden *CustomerID* und *Context* intern von Dialogue Server aufgelöst.

BaseMessageID ist die ID der zu verwendenden Vorlage.

MessageName ist der Name der Nachricht.

ActivityDesc ist die Beschreibung der Aktivität und muss über einen Wert verfügen, wenn ein Aktivitätstyp zum Versenden für die verwendete Vorlage festgelegt wurde.

int64 **ProducePostSingleMessage**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *integer* BaseMessageID, *string* MessageName, *string* ActivityDesc)

Erstellt und speichert eine Nachricht aus einer bestimmten Vorlage für den festgelegten einzelnen Kunden. Die ID der neuen Nachricht wird zurückgegeben.

CustDomainID ist die ID der Kundendomäne. *CustomerID* ist die ID des Kunden, der mit dieser Nachricht in Beziehung steht. *Context ist ein optionaler Kontextwert. Der Wert „0“* bedeutet, dass kein Kontext vorhanden ist.

BaseMessageID ist die ID der zu verwendenden Vorlage.

MessageName ist der Name der Nachricht.

ActivityDesc ist die Beschreibung der Aktivität und muss über einen Wert verfügen, wenn ein Aktivitätstyp zum Versenden für die verwendete Vorlage festgelegt wurde.

string **ProduceSingleDialogMessage**(*string* ServerSession, *integer* CustDomainID, *int64* ParticipantID, *integer* BaseMessageID)

Erstellt eine Nachricht aus einer bestimmten Vorlage für den festgelegten Dialogteilnehmer. Das Ergebnis wird als XML-Dokument zurückgegeben. Die Nachricht wird nicht gespeichert, kann allerdings später mit *PostSingleDialogMessage* gespeichert werden.

CustDomainID ist die ID der Kundendomäne. *ParticipantID* ist die ID des Teilnehmers, mit dem die Nachricht in Beziehung steht.

BaseMessageID ist die ID der zu verwendenden Vorlage.

string **ProduceSingleMessage**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *integer* BaseMessageID)

Erstellt eine Nachricht aus einer bestimmten Vorlage für den festgelegten einzelnen Kunden. Das Ergebnis wird als XML-Dokument zurückgegeben. Die Nachricht wird nicht gespeichert, kann allerdings später mit *PostSingleMessage* gespeichert werden.

CustDomainID ist die ID der Kundendomäne. *CustomerID* ist die ID des Kunden, der mit dieser Nachricht in Beziehung steht. *Context ist ein optionaler Kontextwert. Der Wert „0“* bedeutet, dass kein Kontext vorhanden ist.

BaseMessageID ist die ID der zu verwendenden Vorlage.

string **ProduceSingleTestMessage**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *string* ControlParams, *bool* UsesSql, *string* SqlStatement, *variant* SqlConnectionID, *integer* CustDomainID, *string* CustomerID, *string* Context)

Erstellt eine einzelne Testnachricht auf Basis einer Vorlage und einer Kunden-ID und gibt sie als XML-Dokument zurück.

MessageTypeID ist die eindeutige Kennung des Nachrichtentyps.

ContentText ist der Inhalt der Nachrichtenvorlage, z. B. HTML-Code einer HTML-E-Mailvorlage.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält.

UsesSql gibt an, ob ein benutzerdefinierter SQL-Befehl als Datenquelle beim Zusammenfügen der Nachrichten aus dieser Vorlage verwendet werden soll.

Falls *UsesSQL* den Wert *„true“ hat, muss SqlStatement* den SELECT-Befehl enthalten, der als Datenquelle für die Seriendruck-Nachrichten dienen soll.

Falls *UsesSQL* den Wert „*true“ hat, enthält SqlConnectionID* die ID der Datenbankverbindung, die für die Ausführung des in *SqlStatement* gespeicherten Befehls verwendet wird. Der Wert *null* steht für die Standardverbindung.

CustDomainID ist die ID der Kundendomäne. *CustomerID* ist die ID des Kunden, für den die Testnachricht erstellt wird. *Context* ist ein optionaler Kontextwert; setzen Sie den Wert auf „0“, um keinen Kontext zu verwenden.

string **ProduceTestMessages**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *string* ControlParams, *bool* UsesSql, *string* SqlStatement, *variant* SqlConnectionID, *integer* CustDomainID, *string* FilterExpression, *integer* DialogGroupID, *integer* MaxCount)

Erstellt Testnachrichten auf Basis einer Vorlage und eines Kundensatzes und gibt sie als XML-Dokument zurück.

MessageTypeID ist die eindeutige Kennung des Nachrichtentyps.

ContentText ist der Inhalt der Nachrichtenvorlage, z. B. HTML-Code einer HTML-E-Mailvorlage.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält.

UsesSql gibt an, ob ein benutzerdefinierter SQL-Befehl als Datenquelle beim Zusammenfügen der Nachrichten aus dieser Vorlage verwendet werden soll.

Falls *UsesSQL* den Wert „*true“ hat, muss SqlStatement* den SELECT-Befehl enthalten, der als Datenquelle für die Seriendruck-Nachrichten dienen soll.

Falls *UsesSQL* den Wert „*true“ hat, enthält SqlConnectionID* die ID der Datenbankverbindung, die für die Ausführung des in *SqlStatement* gespeicherten Befehls verwendet wird. Der Wert *null* steht für die Standardverbindung.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

FilterExpression ist ein Ausdruck zur Bestimmung der einzubeziehenden Kunden.

DialogGroupID ist die optionale Angabe der ID einer Gruppe in einem Dialog, welche die zu verwendenden Kunden enthält.

MaxCount ist die Maximalzahl der Kunden, für die Nachrichten erstellt werden. Setzen Sie „MaxCount“ auf -1, um diese Beschränkung der Kundenzahl aufzuheben.

string **ProduceTestMessagesFromSelection**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *string* ControlParams, *bool* UsesSql, *string* SqlStatement, *variant* SqlConnectionID, *integer* CustDomainID, *integer* SelectionID, *integer* MaxCount)

Erstellt Testnachrichten auf Basis einer Vorlage und eines Kundensatzes aus einer Auswahl und gibt sie als XML-Dokument zurück.

MessageTypeID ist die eindeutige Kennung des Nachrichtentyps.

ContentText ist der Inhalt der Nachrichtenvorlage, z. B. HTML-Code einer HTML-E-Mailvorlage.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält.

UsesSql gibt an, ob ein benutzerdefinierter SQL-Befehl als Datenquelle beim Zusammenfügen der Nachrichten aus dieser Vorlage verwendet werden soll.

Falls *UsesSQL* den Wert *„true“ hat, muss SqlStatement* den SELECT-Befehl enthalten, der als Datenquelle für die Seriendruck-Nachrichten dienen soll.

Falls *UsesSQL* den Wert *„true“ hat, enthält SqlConnectionID* die ID der Datenbankverbindung, die für die Ausführung des in *SqlStatement* gespeicherten Befehls verwendet wird. Der Wert *null* steht für die Standardverbindung.

CustDomainID ist die ID der Kundendomäne, auf die zugegriffen wird.

SelectionID ist die ID der Kundenauswahl, die verwendet werden soll.

MaxCount ist die Maximalzahl der Kunden, für die Nachrichten erstellt werden. Setzen Sie „MaxCount“ auf -1, um diese Beschränkung der Kundenzahl aufzuheben.

string **SaveTemplate**(*string* ServerSession, *string* TemplateXML*, bool* SaveAsNew)

Speichert eine Nachrichtenvorlage. Die Methode bekommt als Parameter ein XML-Dokument zur Vorlage und gibt das XML-Dokument mit allen Veränderungen durch Dialogue Server zurück, während sie das Dokument speichert.

TemplateXML ist das XML-Dokument mit der zu speichernden Vorlage.

SaveAsNew gibt an, ob eine Vorlage als neue Vorlage gespeichert oder eine existierende Vorlage aktualisiert werden soll.

void **SendTestMessage**(*string* ServerSession, *integer* MessageTypeID, *string* ContentText, *variant* ContentBinary, *string* ControlParams)

SendTestMessage sendet spontan („On-the-Fly“) eine Nachricht an den bzw. die im Steuerparameter angegebenen Empfänger. Dient zu Testzwecken beim Entwerfen von Nachrichtenvorlagen.

MessageTypeID ist die eindeutige ID des in Dialogue Admin konfigurierten Nachrichtentyps.

ContentText ist der Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die die Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

void **UnlockMessage**(*string* ServerSession, *int64* MessageID)

Entsperrt eine bestimmte Nachricht, die vorher zur Bearbeitung gesperrt wurde. Der Inhalt der Nachricht sollte durch Aufrufen von *UpdateMessage* vor dem Entsperren der Nachricht aktualisiert werden.

MessageID ist die eindeutige ID der zu entsperrenden Nachricht.

void **UpdateMessage**(*string* ServerSession, *int64* MessageID, *string* ContentText, *variant* ContentBinary, *string* ControlParams)

Aktualisiert den Inhalt einer Nachricht.

MessageID ist die eindeutige ID der Nachricht.

ContentText ist der neue Inhalt der Nachricht in dem Fall, dass diese als Text repräsentiert wird.

ContentBinary ist der neue Inhalt der Nachricht in dem Fall, dass diese als ein Array von Bytes binär repräsentiert wird.

ControlParams ist eine Zeichenfolge, die neue Werte für die Steuerparameter enthält. Siehe Abschnitt über Steuerparameter unten.

Steuerparameter

„ControlParams“ wird verwendet, wenn Nachrichten auf elektronischem Weg kommuniziert werden. Die einzelnen Steuerparameter folgen der Syntax:

```
<Parametername>=<Wert>
```

Parameter werden durch Zeilenvorschub (#13#10) getrennt.

Bei einer SMS-Nachricht enthalten die Steuerparameter typischerweise die Telefonnummer des Empfängers. Bei einer E-Mail-Nachricht sind Steuerparameter die Empfängeradresse, der Betreff usw. Im Folgenden ein Beispiel für eine E-Mail:

```
------
Subject=Email message
To address=customer@localhost
From address=demo@localhost
CC address=
BCC address=
Reply address=
Attachments=
Validate target address=True
Priority=Normal
Content transfer encoding=
Content type=
CharSet=
Organization=
Receipt recipient=
----
```

Beispiele

Es sind keine Beispiele verfügbar.

Emarketing Mail-API

Übersicht

API für das Arbeiten mit *Nachrichten-Designer-Vorlagen*. Diese API wird von der Nachrichten-Designer-Webanwendung verwendet.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHEmarketingMailAPI* implementiert die Emarketing Mail-API durch die Schnittstelle *IMHEmarketingMailAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

string **GetBodyArealtemDefinitions**(*string* ServerSession)

Gibt alle Textkörperelemente als XML zurück. Definitionen von Textkörperelementen sind Systemdaten, die bei der Erstellung von Nachrichten-Designer-Vorlagen verwendet werden.

string **GetMasterTemplate**(*string* ServerSession, *integer* MasterTemplateID)

Gibt eine bestimmte Nachrichten-Designer-Mastervorlage als XML zurück.

MasterTemplateID ist die eindeutige Kennung der Mastervorlage.

string **GetPreviewImages**(*string* ServerSession)

Gibt alle verwendeten Vorschaubilder als XML zurück. Vorschaubilder sind Systemdaten, die bei der Erstellung von Nachrichten-Designer-Vorlagen verwendet werden.

string **GetStyleSheetDefs**(*string* ServerSession)

Gibt alle Stylesheet-Definitionen als XML zurück. Stylesheet-Definitionen (CSS-Klassen) sind Systemdaten, die bei der Erstellung von Nachrichten-Designer-Vorlagen verwendet werden.

string **GetTemplate**(*string* ServerSession, *integer* BaseMessageID)

Gibt eine bestimmte Nachrichten-Designer-Vorlage als XML zurück.

BaseMessageID ist die eindeutige Kennung der Nachrichtenvorlage.

string **GetTemplateSchema**(*string* ServerSession)

Gibt das für Nachrichten-Designer-Vorlagen verwendete XML-Schema zurück. Dieses Schema wird als Parameter bzw. Rückgabewert von den Methoden *GetTemplate(...)* und *SaveTemplate(...)* verwendet.

string **SaveTemplate**(*string* ServerSession, *string* TemplateXML, *bool* SaveAsNew)

Speichert eine Emarketing-Nachrichtenvorlage. Die Methode bekommt als Parameter ein XML-Dokument zur Vorlage und gibt das XML-Dokument mit allen Veränderungen durch Dialogue Server zurück, während sie das Dokument speichert.

TemplateXML ist das XML-Dokument mit der zu speichernden Vorlage.

SaveAsNew gibt an, ob eine Vorlage als neue Vorlage gespeichert oder eine existierende Vorlage aktualisiert werden soll.

Beispiele

Es stehen keine Beispiele zur Verfügung.

Quest-API

Übersicht

Die Quest-API bietet eine Reihe von Methoden zum Verwalten von Daten, die mit Fragebögen und Antwortformularen verbunden sind.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHQuestAPI* implementiert die Questionnaire-API über die Schnittstelle *IMHQuestAPI.*

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

void **DeleteAnswerForm**(*string* ServerSession, *int64* AnswerFormID)

Löscht das in *AnswerFormID* festgelegte Antwortformular

integer **DeleteAnswerForms**(*string* ServerSession, *string* SourceXML, *out string* SuccessXML, *out string* ErrorXML)

Löscht Antwortformulare und gibt die Anzahl der erfolgreich gelöschten Antwortformulare zurück.

SourceXML ist das XML-Dokument, das die aus der Datenbank zu löschenden Antwortformulare enthält.

SuccessXML ist ein XML-Dokument, das die erfolgreich aus Dialogue Database gelöschten Antwortformulare enthält.

ErrorXML ist ein XML-Dokument, das die Antwortformulare enthält, die nicht aus der Datenbank gelöscht werden konnten. Jedes Formular in diesem Dokument enthält eine Fehlermeldung mit einer Beschreibung der Ursache für das Fehlschlagen des Löschens.

integer **DeleteAnswerFormsUNC**(*string* ServerSession, *string* SourceUNC, *string* SuccessUNC, *string* ErrorUNC)

Löscht Antwortformulare und gibt die Anzahl der erfolgreich gelöschten Antwortformulare zurück. Entspricht *DeleteAnswerForms*, allerdings werden Dateipfade anstelle von XML-Zeichenfolgen verwendet.

SourceUNC ist die UNC der XML-Datei, welche die aus der Datenbank zu löschenden Antwortformulare enthält.

SuccessUNC ist die UNC der XML-Datei, welche die erfolgreich aus Dialogue Database gelöschten Antwortformulare enthält.

ErrorUNC ist die UNC der XML-Datei, welche die Antwortformulare enthält, die nicht aus der Datenbank gelöscht werden konnten. Jedes Formular in diesem Dokument enthält eine Fehlermeldung mit einer Beschreibung der Ursache für das Fehlschlagen des Löschens.

string **ExportAnswerForms**(*string* ServerSession, *integer* QuestionnaireID, *string* SQLName)

Exportiert Antwortformulare aus Dialogue Database. Die exportierten Antwortformulare werden NICHT aus der Datenbank entfernt. Gibt ein XML-Dokument mit Antwortformularen zurück.

QuestionnaireID schränkt die in einen einzelnen Fragebogen exportierten Antwortformulare ein, wenn ein Wert größer als 0 festgelegt wurde.

SQLName ist ein optionaler Parameter, der sich auf eine SQL-Definition im SQL-Repository (Dialogue Admin) bezieht. Falls angegeben, wird durch diese SQL eine Liste von IDs der zu exportierenden Antwortformulare (qaf_id) zurückgegeben. Die durch diese SQL zurückgegebene Spalte sollte *mh_answer_form_id* benannt werden. Wenn keine SQL angegeben wurde, werden alle Antwortformulare in der Datenbank exportiert.

string **ExportAnswerFormsEx**(*string* ServerSession, *integer* QuestionnaireID, *bool* IncludeComplete, *bool* IncludeUncomplete, *string* SQLName)

Exportiert Antwortformulare aus Dialogue Database. Die exportierten Antwortformulare werden NICHT aus der Datenbank entfernt. Gibt ein XML-Dokument mit Antwortformularen zurück.

QuestionnaireID schränkt die in einen einzelnen Fragebogen exportierten Antwortformulare ein, wenn ein Wert größer als 0 festgelegt wurde.

IncludeComplete legt fest, dass ausschließlich als abgeschlossen markierte Antwortformulare zurückgegeben werden.

IncludeIncomplete legt fest, dass ausschließlich nicht als abgeschlossen markierte Antwortformulare zurückgegeben werden.

Hinweis: Sie können nicht gleichzeitig *IncludeComplete* und *IncludeIncomplete* auf **false** setzen.

SQLName ist ein optionaler Parameter, der sich auf eine SQL-Definition im SQL-Repository (Dialogue Admin) bezieht. Falls angegeben, wird durch diese SQL eine Liste von IDs der zu exportierenden Antwortformulare (qaf_id) zurückgegeben. Die durch diese SQL zurückgegebene Spalte sollte *mh_answer_form_id* benannt werden. Wenn keine SQL angegeben wurde, werden alle Antwortformulare in der Datenbank exportiert.

void **ExportAnswerFormsUNC**(*string* ServerSession, *integer* QuestionnaireID, *string* SQLName, *string* ExportUNC)

Entspricht „ExportAnswerForms“, allerdings werden Dateipfade anstelle von XML-Zeichenfolgen verwendet.

ExportUNC ist die UNC der XML-Datei, welche die exportierten Antwortformulare enthält.

void **ExportAnswerFormsUNCEx**(*string* ServerSession, *integer* QuestionnaireID, *bool* IncludeComplete, *bool* IncludeUncomplete, *string* SQLName, *string* ExportUNC)

Entspricht „ExportAnswerFormsEx“, allerdings werden Dateipfade anstelle von XML-Zeichenfolgen verwendet.

ExportUNC ist die UNC der XML-Datei, welche die exportierten Antwortformulare enthält.

string **GetAnswerForm**(*string* ServerSession, *integer* CustDomainID, *string* CustomerID, *string* Context, *integer* QuestionnaireID)

Gibt eine XML zurück, welche das vom festgelegten Kunden früher beantwortete Antwortformular enthält.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

CustomerID ist die ID des Kunden, während *Context* die verwendeten Kontextwerte beim Versenden der Antwort durch den Kunden sind.

QuestionnaireID ist die ID des Fragebogens.

string **GetAnswerFormByID**(*string* ServerSession*, int64* AnswerFormID)

Gibt eine XML mit dem von einem Kunden früher beantworteten Antwortformular zurück.

AnswerFormID ist die ID des abzurufenden Antwortformulars.

string **GetAnswerFormByIDEx**(*string* ServerSession*, int64* AnswerFormID, *out string* AnswerFormXML, *out string* AnswerFormState)

Ruft eine XML mit dem von einem Kunden früher beantworteten Antwortformular ab.

Gibt die Kennung des Antwortformulars zurück.

AnswerFormID ist die ID des abzurufenden Antwortformulars.

AnswerFormXML enthält die Antwortformular-XML.

AnswerFormState enthält den Status des Antwortformulars, wenn dieses noch nicht abgeschlossen wurde.

string **GetAnswerFormEx**(*string* ServerSession*, integer* CustDomainID, *string* CustomerID, *string* Context, *integer* QuestionnaireID, *out string* AnswerFormXML, *out string* AnswerFormState)

Ruft eine XML mit dem von einem festgelegten Kunden früher beantworteten Antwortformular ab.

Gibt die Kennung des Antwortformulars zurück.

CustDomainID ist die ID, welche die Kundendomäne festlegt.

CustomerID ist die ID des Kunden, während *Context* die verwendeten Kontextwerte beim Versenden der Antwort durch den Kunden sind.

QuestionnaireID ist die ID des Fragebogens.

AnswerFormXML enthält die Antwortformular-XML.

AnswerFormState enthält den Status des Antwortformulars, wenn dieses noch nicht abgeschlossen wurde.

string **GetAnswerFormSchema**(*string* ServerSession)

Gibt eine XML mit der Definition einer Antwortformular-XML zurück. Dies ist sowohl die durch *GetAnswerForm()* und *GetAnswerFormByID()* zurückgegebene XML als auch die XML, die beim Aufrufen von *PostAnswerForm()* zum Dialogue Server gesendet wird.

string **GetQuestionnaire**(*string* ServerSession, *string* QuestionnaireID, *bool* IncludeLayouts)

Gibt ein XML-Dokument mit der formellen Definition eines angegebenen Fragebogens zurück.

IncludeLayouts legt fest, dass alle Layoutdefinitionen des Fragebogens in der XML eingeschlossen werden, wenn *true* ist. Anderenfalls wird lediglich die Definition des Fragebogens mit Abschnitten und Fragen eingeschlossen.

string **GetQuestionnaires**(*string* ServerSession, *string* QuestionnaireID)

Gibt ein XML-Dokument mit einer Liste aller verfügbaren Fragebögen zurück.

datetime **GetQuestionnaireTimestamp**(*string* ServerSession, *string* QuestionnaireID)

Gibt den geänderten Zeitstempel eines Fragebogens zurück, der den letzten Zeitpunkt angibt, an dem die Fragebogendefinition im Fragebogen-Designer geändert wurde.

string **GetResponseTrackLogSchema**(*string* ServerSession)

Gibt das Schema der XML beim Senden eines Antwortnachverfolgungsprotokolls mithilfe der *PostResponseTrackLog( )*-Methode zurück.

string **GetStyle**(*string* ServerSession, *integer* StyleID)

Gibt ein XML-Dokument mit der Definition eines Fragebogens zurück, einschließlich der entsprechenden HTML-Vorlage und CSS-Klassen.

StyleID ist die eindeutige Kennung des Stils.

string **GetStyleSheetDefs**(*string* ServerSession)

Gibt ein XML-Dokument mit einer Liste aller beim Rendering von Fragebögen im Internet verwendeten CSS-Klassen zurück.

integer **ImportAnswerForms**(*string* ServerSession, *string* SourceXML, *out string* SuccessXML, *out string* ErrorXML)

Importiert Antwortformulare und gibt die Anzahl der erfolgreich importierten Antwortformulare zurück.

SourceXML ist das XML-Dokument, das die in Dialogue Database zu importierenden Antwortformulare enthält.

SuccessXML ist das XML-Dokument, das die erfolgreich aus der Datenbank importierten Antwortformulare enthält. Die Antwortformulare erhalten in diesem Dokument neue eindeutige IDs (*answer_form_id*).

ErrorXML ist ein XML-Dokument mit den Antwortformularen, die nicht importiert werden konnten. Jedes Formular in diesem Dokument enthält eine Fehlermeldung mit einer Beschreibung der Ursache des fehlgeschlagenen Importversuchs.

integer **ImportAnswerFormsUNC**(*string* ServerSession, *string* SourceUNC, *string* SuccessUNC, *string* ErrorUNC)

Importiert Antwortformulare und gibt die Anzahl der erfolgreich importierten Antwortformulare zurück.

SourceUNC ist die UNC der XML-Datei, welche die in Dialogue Database zu importierenden Antwortformulare enthält.

SuccessUNC ist die UNC der XML-Datei, welche die erfolgreich aus der Datenbank importierten Antwortformulare enthält. Die Antwortformulare erhalten in diesem Dokument neue eindeutige IDs (*answer_form_id*).

ErrorUNC ist die UNC der XML-Datei, welche die Antwortformulare enthält, die nicht importiert werden konnten. Jedes Formular in diesem Dokument enthält eine Fehlermeldung mit einer Beschreibung der Ursache des fehlgeschlagenen Importversuchs.

int64 ***PostAnswerForm***(*string* ServerSession*, string* AnswerFormXML)

Aktualisiert oder fügt ein Antwortformular ein.

AnswerFormXML ist ein XML-Dokument, welches das neue oder aktualisierte Antwortformular enthält. Wenn das Antwortformular mit einem Kunden verbunden ist, muss die Kunden-ID in der XML enthalten sein. Die XML folgt der Definition der **Antwortformular-XML**, so wie diese von der *GetAnswerFormSchema()*-Methode abgerufen wurde.

int64 ***PostAnswerFormEx***(*string* ServerSession*, string* AnswerFormXML, *string* CustomerDataXML, *string* AnswerFormState)

Aktualisiert oder fügt ein Antwortformular ein.

AnswerFormXML ist ein XML-Dokument, welches das neue oder aktualisierte Antwortformular enthält. Wenn das Antwortformular mit einem Kunden verbunden ist, muss die Kunden-ID in der XML enthalten sein. Die XML folgt der Definition der **Antwortformular-XML**, so wie diese von der *GetAnswerFormSchema()*-Methode abgerufen wurde.

CustomerDataXML

AnswerFormState ist eine Zeichenfolge, in der optional der Status des Antwortformulars enthalten ist. Dies kann beim Speichern eines unvollständigen Antwortformulars verwendet werden.

void **PostResponseTrackLog**(*string* ServerSession, *string Response*TrackLogXML)

Sendet eine Reihe von Nachverfolgungsprotokollelementen zur Datenbank. Antwortnachverfolgungsprotokollelemente werden in der Datenbanktabelle *QRY_RESPONSE_TRACK_LOG* gespeichert.

ResponseWebTrackLogXML ist das XML-Dokument, das die Nachverfolgungsprotokollelemente enthält. Dieses XML-Dokument wird durch das von *GetResponseTrackLogSchema( )* zurückgegebene XML-Schema definiert.

Beispiele

Es sind keine Beispiele verfügbar.

Report-API

Übersicht

Die Report-API bietet eine Reihe von Methoden zum Zugriff auf und Generieren von Berichten.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHReportAPI* implementiert die Report-API über die Schnittstelle *IMHReportAPI.*

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

void **DeleteArchivedReport**(*string* ServerSession, *integer* ReportArchiveID)

Löscht einen Bericht aus dem Archiv.

ReportArchiveID ist die eindeutige ID des archivierten Berichts.

string **FormatBinaryReport**(*string* ServerSession, *string* ReportData, *integer* FormatIndex)

Nimmt ein XML-Dokument mit der binären Darstellung eines Berichts und gibt dieses im angegebenen Format zurück. Die Inhalte der diesem Format entsprechenden Ausgabedateien sind im zurückgegebenen XML-Dokument enthalten.

ReportData ist ein XML-Dokument, das Schlüsseldaten und die binäre Darstellung eines früher erstellten Berichts enthält. *ReportData* ist typischerweise ein durch Aufrufe von *GenerateBinaryReport( )* oder *GenerateReport( )* zurückgegebenes XML-Dokument.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).Siehe* **Berichtsformate** für eine Liste von Formatdefinitionen.

integer **GenerateArchivedReport**(*string* ServerSession, *integer* ReportTemplateID, *string* Params, *string* ReportName)

Erzeugt einen Bericht und speichert das Ergebnis im Berichtsarchiv der Datenbank. Es wird die eindeutige ID des gespeicherten archivierten Berichts zurückgegeben.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall. Das XML-Format wird im Thema **Berichtsparameter-XML** beschrieben.

ReportName ist der Name des Berichts im Archiv.

Hinweis: Verwenden Sie *GetArchivedReport(..)*, um das Ergebnis eines erzeugten Berichts in einem bestimmten Dokumentformat zu erhalten.

string **GenerateBinaryReport**(*string* ServerSession, *integer* ReportTemplateID, *string* Params)

Erzeugt einen Bericht und gibt ein XML-Dokument mit Schlüsseldaten und der binären Darstellung des Berichts zurück.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall. Das XML-Format wird im Thema **Berichtsparameter-XML** beschrieben.

Hinweis: Um das von *GenerateBinaryReport( )* zurückgegebene Ergebnis in ein bestimmtes Ausgabeformat zu konvertieren, rufen Sie *FormatBinaryReport( )* auf.

string **GenerateReport**(*string* ServerSession, *integer* ReportTemplateID, *string* Params, *integer* FormatIndex, *boolean* IncludeBinaryOutput)

Erzeugt einen Bericht im festgelegten Format und gibt das Ergebnis als XML-Dokument zurück. Die Inhalte der diesem Format entsprechenden Ausgabedateien sind im zurückgegebenen XML-Dokument enthalten.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall. Das XML-Format wird im Thema **Berichtsparameter-XML** beschrieben.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).Siehe* **Berichtsformate** für eine Liste von Formatdefinitionen.

IncludeBinaryFormat legt fest, ob die binäre Darstellung des erzeugten Berichts im zurückgegebenen XML-Dokument eingeschlossen ist. Diese binäre Darstellung wird in Aufrufen von *FormatBinaryReport( )* und *SaveReportToArchive( )* benötigt.

void **GenerateReportUNC**(*string* ServerSession, *integer* ReportTemplateID, *string* Params, *integer* FormatIndex, *string* ReportUNC)

Erzeugt einen Bericht im festgelegten Format und speichert diesen in einer Datei.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall. Das XML-Format wird im Thema **Berichtsparameter-XML** beschrieben.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).Siehe* **Berichtsformate** für eine Liste von Formatdefinitionen.

ReportUNC ist der vollständige Pfad und Dateiname zur Zielberichtsdatei.

Hinweis: Auf dem Datenträger wird lediglich die Hauptberichtsdatei gespeichert, d. h. Berichtsformate, die mehrere Dateien erzeugen, werden nicht unterstützt.

string **GetArchivedReport**(*string* ServerSession, *integer ReportArchiveID, integer* FormatIndex)

Gibt einen archivierten Bericht im festgelegten Format als XML-Dokument zurück.

ReportArchiveID ist die eindeutige ID des archivierten Berichts.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).Siehe* **Berichtsformate** für eine Liste von Formatdefinitionen.

string **GetArchivedReportInfo**(*string* ServerSession, *integer* ReportArchiveID)

Gibt Details über einen einzelnen archivierten Bericht als XML zurück.

ReportArchiveID ist die eindeutige ID des archivierten Berichts.

string **GetArchivedReportList**(*string* ServerSession, *integer* ReportTemplateID)

Gibt eine Liste aller archivierten, unter einer Vorlage sortierten Berichte als XML zurück.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

void **GetArchivedReportUNC**(*string* ServerSession, *integer* ReportArchiveID, *integer* FormatIndex, *string* ReportUNC)

Speichert einen archivierten Bericht mit dem festgelegten Format in einer Datei. *ReportArchiveID* ist die eindeutige ID des archivierten Berichts.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).* Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

ReportUNC ist der vollständige Pfad und Dateiname zur Zielberichtsdatei.

Hinweis: Auf dem Datenträger wird lediglich die Hauptberichtsdatei gespeichert, d. h. Berichtsformate, die mehrere Dateien erzeugen, werden nicht unterstützt.

string **GetParamLookupDataset**(*string* ServerSession, *integer* ReportParamID)

Gibt ein XML-Dokument mit Suchwerten eines Berichtsparameters zurück.

string **GetReportFormats**(*string* ServerSession)

Gibt ein XML-Dokument mit allen Berichtsausgabeformaten (alle unterstützten Dokumentformate) zurück. Jedes Format verfügt über einen Formatindex und eine Beschreibung. Die Format-Indexwerte werden bei Aufrufen von anderen Methoden, z. B. *GenerateReport(..)*, verwendet.

Siehe **Berichtsformate** für eine Liste von Formatdefinitionen.

string **GetReportParamSchema**(*string* ServerSession)

Gibt das Schema der XML zurück, die zur Festlegung von Parameterwerten in Aufrufen von Methoden wie „GenerateReport(..)“ und „GenerateArchivedReport(..)“ verwendet wird.

string **GetReportPortalURL**(*string* ServerSession, *integer* ReportTemplateID, *string* Params, *integer* FormatIndex, *string* ViewAction)

Gibt eine URL zurück, die zum Ausführen und Anzeigen eines Berichts in der Report Portal-Anwendung verwendet wird.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

Params ist ein XML-Dokument, das eine Reihe von beim Erzeugen des Berichts verwendeten Parametern enthält. Einige Berichtsvorlagen erfordern die Festlegung einer Reihe von Parametern, bei anderen ist dies nicht der Fall. Das XML-Format wird im Thema **Berichtsparameter-XML** beschrieben.

FormatIndex ist eine Zahl, durch die das gewünschte Dokumentformat des zurückzugebenen Berichts festgelegt wird. Die Format-Zahlenwerte können abgerufen werden durch Aufrufen von: *GetReportDeviceTypes(..).Siehe* **Berichtsformate** für eine Liste von Formatdefinitionen.

ViewAction legt fest, wie der Bericht durch Report Portal angezeigt wird. Siehe **Report View-Aktionen** für eine Liste möglicher Werte.

string **GetReportTemplateInfo**(*string* ServerSession, *integer* ReportTemplateID)

Gibt detaillierte Informationen zu einer Berichtsvorlage als XML zurück. Alle Parameterdefinitionen des Berichts sind eingeschlossen.

ReportTemplateID ist die eindeutige ID der Berichtsvorlage.

string **GetReportTemplates**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle Ordner für Berichtsvorlagen und Berichtsvorlagen enthält.

string **GetSystemReportTemplates**(*string* ServerSession, *string* ReportSystemType)

Gibt ein XML-Dokument zurück, das alle Berichtsvorlagen mit dem festgelegten *Systemberichtstyp* enthält. Ein Systemberichtstyp ist eine optionale Einstellung einer Berichtsvorlage, der beschreibt, über welchen „object“-Typ ein Bericht erstellt wird. Alle Systemberichtstypen werden in der Datenbanktabelle REPORT_SYSTEM_TYPE definiert.

integer **SaveReportToArchive**(*string* ServerSession, *string* ReportData, *string* ReportName)

Speichert einen in „ReportData“ enthaltenen Bericht in das Berichtsarchiv. Die Archiv-ID wird zurückgegeben.

ReportData ist ein XML-Dokument, das Schlüsseldaten und die binäre Darstellung eines früher erstellten Berichts enthält. *ReportData* ist typischerweise ein durch Aufrufe von *GenerateBinaryReport( )* oder *GenerateReport( )* zurückgegebenes XML-Dokument.

Beispiele

Im folgenden Beispiel wird vom Dialogue Server ein Bericht generiert und das Ergebnis als ein XML-Dokument zurückgegeben. Wenn beispielsweise der generierte Bericht im PDF-Format vorliegt, enthält die XML das codierte binäre PDF-Dokument. Das Beispiel verwendet JScript.

```
  ..............
  //Connect to the DialogServer
  SystemAPI = new ActiveXObject("MHDialogServer.MHReportAPI");

  //Log in
  var InstanceName = "MHProduction";
  var SessionKey = SystemAPI.Login(InstanceName,
"admin", "ringo1", "MH Test");
  var ServerSession = SessionKey + "@"
+ InstanceName;

  //Now generate a report based on a report template, and
retrieve the report output
  ReportAPI = new ActiveXObject("MHDialogServer.MHReportAPI");
  var ReportXML = ReportAPI.GenerateReport(ServerSession,
1000, "", 3, False);

  //Log out
  SystemAPI.Logout(ServerSession);
```

. . . . . .

Selection-API

Übersicht

Die Selection-API bietet eine Reihe von Methoden zum Zugriff auf und zur Verwaltung von Auswahlen.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHSelectionAPI* implementiert die Dialogue-API über die Schnittstelle *IMHSelectionAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

void **DeleteCustomerList**(*string* ServerSession, *integer* ListID)

Löscht eine vorhandene Kundenliste.

ListID ist die ID einer vorhandenen Liste (Datenbankspalte `LIST.LST_ID`).

void **DeletePsrCustomerList**(*string* ServerSession, *integer* PsrListID)

Löscht eine vorhandene Kundenliste, die von der ID identifiziert wird, die sie im Portrait Shared Repository hat. Es wird eine Ausnahme ausgegeben, wenn keine Liste mit dem angegebenen Parameter *PsrListID* gefunden wird.

PsrListID ist die PSR-ID einer im Portrait Shared Repository (PSR) vorhandenen Liste.

void **DeletePsrSelection**(*string* ServerSession, *integer* PsrSelectionID)

Löscht eine vorhandene PD-Auswahl, die von der ID identifiziert wird, die sie im Portrait Shared Repository hat. Es wird eine Ausnahme ausgegeben, wenn keine Auswahl mit dem angegebenen Parameter *PsrSelectionID* gefunden wird.

PsrSelectionID ist die ID einer im PSR vorhandenen Auswahl.

void **DeleteSelection**(*string* ServerSession, *integer* SelectionID)

Löscht eine vorhandene Auswahl.

SelectionID entspricht der ID der Auswahl.

string **GetCustomerLists**(*string* ServerSession)

Gibt ein XML-Dokument mit Beschreibungen aller definierten Kundenlisten für alle Kundendomänen zurück.

Hinweis: *GetCustomerLists(..)* gibt nicht die Listenmitglieder der Liste zurück.

string **GetSelections**(*string* ServerSession)

Gibt ein XML-Dokument mit allen definierten Kundenauswahlen für alle Kundendomänen zurück.

int **SavePsrCustomerList**(*string* ServerSession, *integer* PsrListID, *integer* DomainID, *string* Name, *string* Description, *string* CompressedListMembers)

Speichert (erstellt oder aktualisiert) eine PD-Liste, die auf einer PSR-Liste basiert (z. B. von Portrait Explorer). Wenn keine Liste vorhanden ist, wird sie erstellt. Ist die Liste bereits vorhanden, werden sie und ihre Listenmitglieder aktualisiert.

SavePsrCustomerList(…) gibt die eindeutige ID der gespeicherten Liste zurück (Datenbankspalte `LIST.LST_ID`).

PsrListID ist die ID der Liste im PSR.

DomainID ist die Domänen-ID von Portrait Dialogue.

Name ist der Name der Liste.

Description ist die Beschreibung der Liste.

CompressedListMembers ist eine Base-64-codierte Zeichenfolge, die einen komprimierten Satz an Kunden-IDs in einem von PSR definierten Format enthält.

int **SavePsrCustomerListAsynchronous**(*string* ServerSession, *integer* PsrListID, *integer* DomainID, *string* Name, *string* Description, *string* CompressedListMembers)

Speichert (erstellt oder aktualisiert) asynchron eine PD-Liste, die auf einer PSR-Liste basiert (z. B. von Portrait Explorer).

Diese Methode ist identisch mit *SavePSrCustomerList(...)*, außer dass sie den Vorgang asynchron ausführt. Sie gibt allerdings nicht die eindeutige ID der gespeicherten Liste zurück, sondern die `SystemTaskID` der Portrait Dialogue-Hintergrundaufgabe, die zur Ausführung dieser Methode erstellt wurde.

Eine Beschreibung der Methodenparameter finden Sie unter *SavePSrCustomerList(…)*.

integer **SavePsrSelection**(*string* ServerSession, *integer* PsrSelectionID, *integer* DomainID, *string* Name, *string* Description, *string* PsrSelectionXml)

Speichert (erstellt oder aktualisiert) asynchron eine PD-Auswahl, die auf einer PSR-Auswahl basiert (z. B. von Portrait Explorer). Wenn keine Auswahl vorhanden ist, wird sie erstellt. Wenn sie vorhanden ist, wird die Auswahl aktualisiert. Gibt die ID der PD-Auswahl zurück.

PsrSelectionID ist die ID der Auswahl im PSR.

DomainID ist die ID der Kundendomäne.

Name ist der Name der Auswahl.

Description ist die Beschreibung der Auswahl.

PsrSelectionXml, XML der PSR-Auswahl.

integer **SaveSelection**(*string* ServerSession, *integer* SelectionID, *integer* DomainID, *string* Name, *string* Description, *string* Expression, *string* ContextExpression, *bool* UseInWebSearch, *bool* UseInTestingMessages, *bool* UseInMessageConditions)

Speichert (erstellt oder aktualisiert) eine systemeigene PD-Auswahl auf Basis der PD-Ausdruckssyntax. Wenn keine Auswahl vorhanden ist, wird sie erstellt. Ist eine Auswahl vorhanden, wird sie aktualisiert. Gibt die ID der Auswahl zurück.

SelectionID ist die ID einer vorhandenen, zu aktualisierenden Auswahl. Geben Sie für eine neue Auswahl einen Wert von *-1* an.

DomainID ist die ID der Kundendomäne.

Name ist der Name der Auswahl.

Description ist die Beschreibung der Auswahl.

Expression ist der Auswahlausdruck.

ContextExpression ist der Kontextausdruck der Auswahl. Wird er leer gelassen, unterstützt die Auswahl keinen Kontext.

Wenn *UseInWebSearch* auf „true“ gesetzt wird, ist die Auswahl in Customer View verfügbar.

Wenn *UseInTestingMessages* auf „true“ gesetzt wird, ist die Auswahl beim Testen von Nachrichtenvorlagen verfügbar.

Wenn *UseInMessageConditions* auf „true“ gesetzt wird, ist die Auswahl beim Definieren von Bedingungen von Nachrichtenvorlagen verfügbar.

integer **SaveSqlSelection**(*string* ServerSession, *integer* SelectionID, *integer* DomainID, *string* Name, *string* Description, *string* Sql, *bool* HasContext, *bool* UseInWebSearch, *bool* UseInTestingMessages, *bool* UseInMessageConditions)

Speichert (erstellt oder aktualisiert) eine systemeigene PD-Auswahl auf Basis einer SQL-Anweisung. Wenn keine Auswahl vorhanden ist, wird sie erstellt. Ist eine Auswahl vorhanden, wird sie aktualisiert. Gibt die ID der Auswahl zurück.

SelectionID ist die ID einer vorhandenen, zu aktualisierenden Auswahl. Geben Sie für eine neue Auswahl einen Wert von *-1* an.

DomainID ist die ID der Kundendomäne.

Name ist der Name der Auswahl.

Description ist die Beschreibung der Auswahl.

Sql ist die SQL-Anweisung der Auswahl.

Wenn *HasContext* auf „true“ gesetzt wird, muss die SQL-Anweisung der Auswahl eine Kontextspalte zurückgeben.

Wenn *UseInWebSearch* auf „true“ gesetzt wird, ist die Auswahl in Customer View verfügbar.

Wenn *UseInTestingMessages* auf „true“ gesetzt wird, ist die Auswahl beim Testen von Nachrichtenvorlagen verfügbar.

Wenn *UseInMessageConditions* auf „true“ gesetzt wird, ist die Auswahl beim Definieren von Bedingungen von Nachrichtenvorlagen verfügbar.

Beispiele

Es stehen keine Beispiele zur Verfügung.

System-API

Übersicht

Die System-API stellt eine Reihe von Methoden auf Systemebene zur Verfügung. Dies sind Funktionalitäten auf der technischen Ebene von Dialogue Server, wie Instanzen, Benutzer und Anmeldungen. Sie bezieht sich nicht auf kunden- oder dialogbezogene Daten.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHSystemAPI* implementiert die System-API über die Schnittstelle *IMHSystemAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

bool **CheckSystemObjectLock**(*string* ServerSession, *string* ObjectName, ***integer*** ObjectID, *out string* LockedByUser)

Prüft, ob ein Systemobjekt gesperrt ist und gibt in diesem Fall *true* zurück. Sperrungen werden normalerweise benutzt, wenn ein Objekt (wie beispielsweise ein Dialog) bearbeitet wird.

ObjectName ist der Name des Objekttyps (definiert in der Datenbanktabelle SYSTEM_OBJECT). Ein Beispiel für einen Objektnamen ist: dialog.

ObjectID ist die eindeutige Kennung der Objektinstanz, wie beispielweise *DialogID*.

LockedByUser ist ein Ausgabeparameter, der den Benutzernamen des Benutzers angibt, der das Systemobjekt sperrt.

Siehe auch *UnlockSystemObject(...)* und *LockSystemObject(...).*

string **GetDefaultUserSetting**(*string* ServerSession)

Gibt ein XML-Dokument mit standardmäßigen Benutzereinstellungen für das aktuelle Anwendungssystem zurück. Die Standardbenutzereinstellungen sind in der Datenbanktabelle APPLICATION_SYSTEM gespeichert.

string **GetInstances**()

Gibt ein XML-Dokument mit allen Instanzen von Dialogue Database innerhalb des angesprochenen *Dialoghost* zurück. Ein Dialoghost ist ein Computer oder Server, auf dem Dialogue Server läuft.

string **GetParameterCollection**(*string* ServerSession, *string* CollectionName)

Gibt ein XML-Dokument mit den Systemparametern, den Parameternamen und -werten zurück. Diese werden in Dialogue Admin definiert.

CollectionName ist der Name der Parametersammlung, die zurückgegeben wird. Systemparameter sind in *Sammlungen* genannte Gruppen unterteilt. Diese werden in Dialogue Admin definiert.

string **GetServerDefaultLanguage**( )

Gibt eine Zeichenfolge mit dem Standardsprachcode von Dialogue Server zurück.

Hinweis: Die Standardsprache kann auf Instanzenebene überschrieben werden. Die XML, die von *GetInstances(...)* zurückgegeben wird, enthält den Sprachcode jeder Instanz.

string **GetServerName**( )

Gibt eine Zeichenfolge mit dem Namen des Computers zurück, auf dem Dialogue Server ausgeführt wird.

string **GetServerVersion**( )

Gibt eine Zeichenfolge mit der Versionsnummer von Dialogue Server zurück. Zum Beispiel „4.4.0.5“.

string **GetSystemUserInfos**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle vom System definierten Systembenutzer enthält. Systembenutzer werden in Dialogue Admin hinzugefügt und konfiguriert. Das Dokument enthält auch Informationen über die Verbindung zwischen Benutzer und Aufgabenarbeitsgruppe (twgm_twg_id).

string **GetUserSessionInfo**(*string* ServerSession)

Gibt ein XML-Dokument mit Informationen über die aktuelle Anmeldung oder Benutzersitzung zurück. Dies beinhaltet Zugriffsrechte, Eigentümerschaft an Objekten und angepasste Einstellungen.

string **GetUserSessionInfoSimple**(*string* ServerSession)

Gibt ein XML-Dokument mit grundlegenden Informationen über die aktuelle Anmeldung oder die Benutzersitzung zurück. Diese Methode gibt einen Teil der Informationen zurück, die von *GetSystemUserInfo(...)* geliefert werden, was Zugriffsrechte, Eigentümerschaft an Objekten und angepasste Einstellungen nicht umfasst.

void **LockSystemObject**(*string* ServerSession, *string* ObjectName, ***integer*** ObjectID **)**

Sperrt ein Systemobjekt im Kontext des aktuellen Benutzers. Sperrungen werden normalerweise benutzt, wenn ein Objekt (wie beispielsweise ein Dialog) bearbeitet wird.

ObjectName ist der Name des Objekttyps (definiert in der Datenbanktabelle SYSTEM_OBJECT). Ein Beispiel für einen Objektnamen ist: dialog.

ObjectID ist die eindeutige Kennung der Objektinstanz, wie beispielweise *DialogID*.

Siehe auch *UnlockSystemObject(...)* und *CheckSystemObjectLock(...).*

string **Login**(*string* InstanceName, *string* UserName, *string* Password, *string* ApplicationSystem)

Meldet einen Benutzer auf dem Dialogue Server an. Es werden seine Serversitzung und eine eindeutiger Zeichenfolge (ein Sitzungsschlüssel) erstellt, der zurückgegeben wird und diese Sitzung identifiziert.

InstanceName ist der Name der Dialogue Database-Instanz, auf die zugegriffen wird.

UserName und *Password* identifizieren den sich anmeldenden Benutzer.

ApplicationSystem ist das Anwendungssystem, das von Dialogue Server zur Sitzungserstellung autorisiert ist. Anwendungssysteme werden in Dialogue Admin eingerichtet.

Hinweis: Das Aufrufen der meisten Methoden der Dialogue Server-API erfolgt mit dem Zeichenfolgenargument *ServerSession.* Diese Zeichenfolge besteht aus der von *Login* zurückgegebenen Kennung plus dem Instanzenname. Das Format von „ServerSession“ ist:

```
<session key>@<instance name>
```

string **LoginDelegate**(*string* InstanceName, *string* DelegatorUserName, *string* DelegatorPassword, *string* ApplicationSystem, *string* UserName, *bool* RequireWinAuthEnabled)

Lässt einen stellvertretenden Benutzer einen anderen Benutzer am Dialogue Server anmelden. Für den angegebenen Benutzer wird eine Sitzung erstellt und der Sitzungsschlüssel wird zurückgegeben.

InstanceName ist der Name der Dialogue Database-Instanz, auf die zugegriffen wird.

DelegatorUserName und *DelegatorPassword* identifizieren den Benutzer, der angemeldet wird.

ApplicationSystem ist das Anwendungssystem, das von Dialogue Server zur Sitzungserstellung autorisiert ist. Anwendungssysteme werden in Dialogue Admin eingerichtet.

UserName ist der Benutzername des Benutzers, der angemeldet wird.

RequireWinAuthEnabled entscheidet, ob der anzumeldende Benutzer die Windows-Authentifizierung aktiviert haben muss.

Hinweis: Der delegierte Benutzer muss das Recht **Erstellung von Benutzersitzungen für andere Benutzer durch den Benutzer zulassen** besitzen.

string **LoginEx**(*string* ServerSession, *string* ApplicationSystem)

Diese Methode ist identisch mit *Login*, nur dass der Parameter „ServerSession“ einen bereits gültigen Sitzungsschlüssel angibt, anstelle der Angabe vom Instanzen- und Benutzername sowie Kennwort. *LoginEx* ermöglicht die Anmeldung aus einem anderen Anwendungssystem, an dem der Benutzer bereits angemeldet ist.

string **LoginWinAuth**(*string* InstanceName, *string* ApplicationSystem)

Diese Methode ist identisch mit *Login*, nur dass der Benutzer mit der Windows-Authentifizierung identifiziert wird. Der Dialogue Server verwendet die ID des Anrufers, um den Benutzer zu identifizieren und überprüft dann, ob der Benutzer existiert und mit der Windows-Authentifizierung konfiguriert ist.

string **Logout**(*string* ServerSession)

Meldet die angegebene Serversitzung ab.

void **SetUserSessionLanguage**(*string* ServerSession, *string* LanguageCode)

Legt die Sprache der aktuellen Benutzersitzung fest. Dadurch gibt Dialogue Server Fehlermeldungen und andere Zeichenfolgen in der angegebenen Sprache zurück.

LanguageCode ist der Code der Sprache, wie beispielsweise „en-US“ (US-amerikanisches Englisch) oder „de-DE“ (Französisch).

Hinweis: Die Softwaremodule (wie beispielsweise Dialogue Server) unterstützen mehrere Sprachen. Die Zeichenfolgen in der Datenbank hingegen sind nicht mehrsprachig, sondern in der Sprache, die bei der Installation oder der Aktualisierung gewählt wurde.

void **UnlockSystemObject**(*string* ServerSession, *string* ObjectName, *integer* ObjectID)

Entsperrt ein Systemobjekt.

ObjectName ist der Name des Objekttyps (definiert in der Datenbanktabelle SYSTEM_OBJECT). Ein Beispiel für einen Objektnamen ist: dialog.

ObjectID ist die eindeutige Kennung der Objektinstanz, wie beispielweise *DialogID*.

Siehe auch *LockSystemObject(...)* und *CheckSystemObjectLock(...).*

void **UpdateUserSettings**(*string* ServerSession, *string* Settings)

Speichert eine Zeichenfolge namens *Settings* in Dialogue Database. Diese Zeichenfolge kann beispielsweise ein XML-Dokument sein. Sie wird auf der Ebene einzelner *Benutzer* abgespeichert und ist Teil des XML-Dokuments, das die Methode *GetUserSessionInfo* zurückgibt.

bool **ValidateServerSession**(string ServerSession)

Überprüft eine Serversitzung und gibt „true“ zurück, wenn sie gültig ist. Anderenfalls wird „false“ zurückgegeben.

Beispiele

Im folgenden Beispiel werden eine Serversitzung erstellt und Informationen über einige Kunden abgerufen, die vom Dialogue Server geliefert werden. Das Beispiel verwendet JScript.

```
..............
//Connect to the DialogServer
SystemAPI = new ActiveXObject("MHDialogServer.MHSystemAPI");

//Log in
var InstanceName = "MHProduction";
var SessionKey = SystemAPI.Login(InstanceName, "admin", "ringo1", "MH Test");
var ServerSession = SessionKey + "@" + InstanceName;

//Now get some customers
CustomerAPI = new ActiveXObject("MHDialogServer.MHCustomerAPI");
var CustomerXML = CustomerAPI.GetCustomers(ServerSession, 1000, "Address.Ci-
ty=\"Oslo\"", "", "", -1);

//Log out (we got what we wanted)
SystemAPI.Logout(ServerSession);

..............
```

Telemarketing-API

Übersicht

Die Telemarketing-API bietet eine Reihe von Methoden zum Zugriff auf Telemarketing-Projekte und -Teilnehmer.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHTelemarketingAPI* implementiert die Telemarketing-API über die Schnittstelle *IMHTelemarketingAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

int64 **CallComplete**(*string* ServerSession, *integer* CCProjectID, *int64* ParticipantID, *string* CallStatus, *datetime* NextCallDateTime, *string* AssignToUserName, *string* Note, *string* AnswerFormDataXML)

Schließt eine einzelne Aufrufoperation ab, indem dem Telemarketing-Teilnehmer ein neuer Status gegeben wird. Wenn vom Telemarketing-Projekt ein Fragebogen verwendet wird, sendet diese Methode zudem das durch den Parameter *AnswerFormDataXML* festgelegte Antwortformular. Wenn ein Formular gesendet wird, ist der Rückgabewert von *CallComplete* die ID des Antwortformulars. Anderenfalls wird -1 zurückgegeben.

CCProjectID ist die ID des Telemarketing-Projekts.

ParticipantID ist die ID des Teilnehmers. Dies ist die Dialogteilnehmer-ID.

CallStatus ist der technische Name des als aktuellen Status des Telemarketing-Teilnehmers festzulegenden Aufrufstatus. Aufrufstatustypen werden in Dialogue Admin definiert.

AssignToUserName ist der Name des Benutzers, der den Teilnehmer zu einem späteren Zeitpunkt anrufen wird. Wenn „AssignToUserName“ leer gelassen wird, kann jeder beliebige Telemarketing-Benutzer diesen Teilnehmer abrufen.

Note ist ein im Anrufverlauf des Teilnehmers gespeicherter Textwert.

AnswerFormDataXML ist ein XML-Dokument, welches das für den Abschluss der Telemarketing-Operation benötigte Antwortformular enthält. Das XML-Dokument entspricht der in Microsoft ADO.NET definierten Form eines XML-DiffGrams. Ein Telemarketing-Projekt kann oder kann nicht über einen verbundenen Fragebogen verfügen.

string **FetchNextToCall**(*string* ServerSession, *integer* CCProjectID, *string* FilterExpression, *string* AssignedToUserName, *string* CurrentCallStatus, *bool* FilterOnRecall, *string* DataFields)

Gibt ein XML-Dokument mit dem nächsten anzurufenden Telemarketing-Teilnehmer zurück. Die XML gibt Kundendaten entsprechend der Festlegung in *DataFields* zurück. Der Teilnehmer hat den Status *locked* in Bezug auf einen Anruf durch den Server.

CCProjectID ist die ID des Telemarketing-Projekts.

FilterExpression ist ein Ausdruck, der zum Filtern oder Verfeinern der zurückgegebenen Teilnehmer aus der Teilnehmerauswahl verwendet wird.

Wenn *AssignedToUserName* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer ausgewählt, die bereits früher dem angegebenen Benutzer zugeordnet wurden.

Wenn *CurrentCallStatus* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer mit dem angegebenen Anrufstatus ausgewählt.

Wenn *FilterOnRecall* auf „true“ festgelegt wurde, werden durch den Server ausschließlich Teilnehmer für erneute Anrufe ausgewählt.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

string **FetchToCall**(*string* ServerSession, *integer* CCProjectID, *int64* ParticipantID, *string* DataFields)

Gibt ein XML-Dokument mit einem festgelegten, anzurufenden Telemarketing-Teilnehmer zurück. Die XML gibt Kundendaten entsprechend der Festlegung in *DataFields* zurück. Der Teilnehmer hat den Status *locked* in Bezug auf einen Anruf durch den Server. Wenn der Teilnehmer in der Anrufwarteschlange nicht vorhanden ist, wird eine Fehlermeldung ausgegeben.

CCProjectID ist die ID des Telemarketing-Projekts.

ParticipantID ist die ID des zurückzugebenen Teilnehmers. Dies ist die Dialogteilnehmer-ID.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

string **GetCallStatusTypes**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle in Dialogue Admin definierten Anrufstatustypen enthält.

integer **GetCCParticipantCount**(*string* ServerSession, *integer* CCProjectID, *string* FilterExpression, *string* LockedByUserName, *string* AssignedToUserName, *string* CurrentCallStatus*, bool* FilterOnRecall)

Gibt die Anzahl der Telemarketing-Teilnehmer zurück, welche die vorhandenen Kriterien erfüllen.

CCProjectID ist die ID des Telemarketing-Projekts.

FilterExpression ist ein Ausdruck zum Filtern oder Verfeinern der Teilnehmerauswahl, aus der die Anzahl ermittelt wird.

Wenn *LockedByUserName* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer ausgewählt, die vom angegebenen Benutzer gesperrt wurden.

Wenn *AssignedToUserName* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer ausgewählt, die bereits früher dem angegebenen Benutzer zugeordnet wurden.

Wenn *CurrentCallStatus* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer mit dem angegebenen Anrufstatus ausgewählt.

Wenn *FilterOnRecall* auf „true“ festgelegt wurde, werden durch den Server ausschließlich Teilnehmer für erneute Anrufe ausgewählt.

string **GetCCParticipantDetails**(*string* ServerSession, *integer* CCProjectID, *int64* ParticipantID)

Gibt ein XML-Dokument mit dem Anrufprotokoll des angegebenen Telemarketing-Teilnehmers zurück.

CCProjectID ist die ID des Telemarketing-Projekts.

ParticipantID ist die ID des zurückzugebenen Teilnehmers. Dies ist die Dialogteilnehmer-ID.

string **GetCCParticipants**(*string* ServerSession, *integer* CCProjectID, *string* FilterExpression, *string* LockedByUserName, *string* AssignedToUserName, *string* CurrentCallStatus, *bool* FilterOnRecall, *string* DataFields, *integer* MaxCount)

Gibt ein XML-Dokument mit Telemarketing-Teilnehmern zurück, die den angegebenen Kriterien entsprechen. Die XML gibt Kundendaten entsprechend der Festlegung in *DataFields* zurück.

CCProjectID ist die ID des Telemarketing-Projekts.

FilterExpression ist ein Ausdruck, der zum Filtern oder Verfeinern der zurückgegebenen Teilnehmer aus der Teilnehmerauswahl verwendet wird.

Wenn *LockedByUserName* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer ausgewählt, die vom angegebenen Benutzer gesperrt wurden.

Wenn *AssignedToUserName* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer ausgewählt, die bereits früher dem angegebenen Benutzer zugeordnet wurden.

Wenn *CurrentCallStatus* festgelegt wurde, werden durch den Server ausschließlich Teilnehmer mit dem angegebenen Anrufstatus ausgewählt.

Wenn *FilterOnRecall* auf „true“ festgelegt wurde, werden durch den Server ausschließlich Teilnehmer für erneute Anrufe ausgewählt.

DataFields ist eine semikolongetrennte Liste von Kundendatengruppen und -feldern (aus der Kundendomänendefinition), die im zurückgegebenen XML-Dokument enthalten sein sollen.

string **GetCCProjects**(*string* ServerSession)

Gibt ein XML-Dokument zurück, das alle definierten Telemarketing-Projekte enthält und beschreibt.

string **GetStatistics**(*string* ServerSession, *integer* CCProjectID)

Gibt ein XML-Dokument zurück, das aggregierte Daten über das angegebene Telemarketing-Projekt enthält.

void **UnlockCCParticipant***(string* ServerSession, *integer* CCProjectID, *int64* ParticipantID)

Entsperrt den angegebenen Telemarketing-Teilnehmer. Dadurch wird ein angefangener Anruf abgebrochen und der Teilnehmer in der Anrufwarteschlange zurückgesetzt. Der entsperrte Teilnehmer muss vorher gesperrt worden sein mittels *FetchNextToCall oder FetchToCall*.

CCProjectID ist die ID des Telemarketing-Projekts.

ParticipantID ist die ID des zu entsperrenden Teilnehmers. Dies ist die Dialogteilnehmer-ID.

Beispiele

Es stehen keine Beispiele zur Verfügung.

Web Utilities-API

Übersicht

Eine API, um auf die Funktionen von Webdienstprogrammen, die E-Mail- und Verknüpfungsnachverfolgung beinhalten, sowie auf Dateien zuzugreifen, die im Internet veröffentlicht wurden.

COM-Komponente

Die COM-Komponente *MHDialogServer.MHWebUtilsAPI* implementiert die Web Utilities-API durch die Schnittstelle *IMHWebUtilsAPI*.

Methoden

Die folgenden Methoden und SOAP-Operationen werden unterstützt:

string **AddWebPublicFile**(*string* ServerSession, *variant* ContentBinary, *string* Filename, *string* Description)

Veröffentlicht eine Datei in Dialogue Server, so dass auf diese durch eine URL zur Anwendung „Web Utilities“ zugegriffen werden kann. Dateien, die für das Internet veröffentlicht wurden, werden in der Datenbanktabelle *WEB_PUBLIC_FILE* gespeichert.

AddWebPublicFile(...) gibt eine *Inhalts-ID* zurück, die automatisch der zu veröffentlichenden Datei zugewiesen wird. Eine Inhalts-ID ist eine Zeichenfolge, welche die Datei eindeutig identifiziert (GUID).

ContentBinary ist der Inhalt der Datei, binär dargestellt oder als Byte-Array.

Filename ist der Name der Datei und der Erweiterungsteile. Pfadinformationen sind davon ausgeschlossen. *Beispiel:* meinlogo.jpg.

Description ist eine optionale Dateibeschreibung, die in Dialogue Database mit der Datei gespeichert wird.

void **DeleteWebPublicFile**(*string* ServerSession, *string* ContentID)

Löscht eine Datei, die im Internet veröffentlicht wurde.

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

string **ExecuteContentObjec**(*string* ServerSession, *string* ContentObjectKey, *string* ContentOutputMode, *integer* CustDomainID, *string* CustomerID, *string* Context, *int64* ParticipantID, *string* RequestParams)

Führt ein Inhaltsobjekt aus und gibt den resultierenden Inhalt zurück.

ContentObjectKey ist eine eindeutige Zeichenfolge, die das Inhaltsobjekt identifiziert. Diesen Wert findet man unter „Inhaltsobjekteigenschaften“ in Visual Dialogue.

ContentOutputMode gibt an, wie der resultierende Inhalt im HTML-Kontext dargestellt werden soll. Mögliche Werte sind:

Wert	Beschreibung
unwrapped	Der reine Inhalt des resultierenden Inhaltselements wird zurückgegeben. Beispiel: Wenn das resultierende Inhaltselement ein Bild ist, wird nur die URL zum Bild zurückgegeben.
wrapped	Das resultierende Inhaltselement wird innerhalb eines passenden HTML-Elements für die Präsentation auf einer HTML-Seite ausgegeben. Beispiel: Wenn das resultierende Inhaltselement ein Bild ist, wird ein <IMG>-Element zurückgegeben.
Seite	Das resultierende Inhaltselement wird in eine HTML-Seite eingefügt. Dieser Modus ist geeignet, wenn das Inhaltsobjekt unter Verwendung eines <IFRAME>-Elements auf einer HTML-Seite präsentiert wird. Beispiel: Wenn das resultierende Inhaltselement ein Bild ist, wird die Quelle einer HTML-Seite mit einem <IMG>-Element zurückgegeben.

CustDomainID, CustomerID und Context identifizieren den Kunden, für den das Inhaltsobjekt ausgeführt wird. Wenn der Benutzergruppentyp des Inhaltsobjekts *identifiziert* ist, wird „CustDomainID“ vergeben, und es muss nicht angegeben werden beim Aufrufen von: *ExecuteContentObject(...).* (Setzen Sie CustDomainID auf „-1“).

Alternativ zur Angabe von „CustomerID“ und „Context“ kann die eindeutige ID eines Dialogteilnehmers angegeben werden: *ParticipantID*.

Falls das Inhaltsobjekt im anonymen Modus aufgerufen wird, setzen Sie „CustDomainID“ sowie „ParticipantID“ auf „-1“, und lassen Sie „CustomerID“ und „Context“ leer.

RequestParams ist eine Zeichenfolge, die zusätzliche Parameter beim Aufrufen der Inhaltsobjekte enthält. Das Format dieser Zeichenfolge ist:

```
<param1>=<value1>;<param2>=<value2>;...;<paramN>=<valueN>
```

Beispiel:

```
MyInfo1=B;MyInfo2=C
```

Hinweis: Anstelle dieser API-Methode ist es möglich, ein Inhaltsobjekt unter Verwendung einer URL über die Anwendung „Web Utilities“ auszuführen. Diese URL kann direkt im HTML-Code einer Webseite verwendet werden. Siehe auch **Inhaltsobjekt-URLs** auf Seite 100

string **GetContentObjects**(*string* ServerSession)

Gibt eine Liste aller aktiven Inhaltsobjekte als XML zurück. Die Schlüsseleigenschaften jedes Inhaltsobjekts sind in der XML enthalten.

string **GetWebPublicFile**(*string* ServerSession, *string* ContentID)

Gibt den Inhalt einer Datei, die im Internet veröffentlicht wurde, als XML zurück.

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

string **GetWebPublicFiles**(*string* ServerSession)

Gibt eine Liste aller im Internet veröffentlichten Dateien als XML zurück. Die Liste enthält Schlüsselinformationen über jede Datei. Die Liste ist nach Dateinamen sortiert.

string **GetWebPublicFileURL**(*string* ServerSession, *string* ContentID)

Gibt die URL einer Datei zurück, die im Internet veröffentlicht wurde. Die URL verweist auf die Anwendung „Web Utilities“.

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

string **GetWebPublicFileURLEx**(*string* ServerSession, *string* ContentID, *integer* RefreshMode)

Bewirkt dasselbe wie *GetWebPublicFileURL(...)*. Allerdings werden zusätzliche Parameter für die Steuerung der Dateizwischenspeicherfunktion von der Anwendung „Web Utilities“ benötigt.

RefreshMode steuert, ob jede zwischengespeicherte Kopie der Datei mit über die URL zurückgegebenen Signalen aktualisiert wird. Mögliche Werte für *RefreshMode* sind:

Wert	Wertname	Beschreibung
0	return	Die Datei ist in der Antwort enthalten, aber der Cache wird nicht aktualisiert.
1	aktualisieren	Der Cache wird gelöscht, aber die Datei ist nicht in der Antwort enthalten.
2	refresh_return	Der Cache wird aktualisiert, und die Datei ist in der Antwort enthalten.

string **GetWebTrackLinkNames**(*string* ServerSession)

Gibt eine Liste der verwendeten Nachverfolgungsnamen für Verknüpfungen als XML zurück.

Hinweis: Die Liste der Verknüpfungsnamen wird in der Datenbank gewartet (WEB_TRACK_LINK_NAME) und vom Server gefüllt, wenn Nachrichten zusammengefügt werden.

string **GetWebTrackLogSchema**(*string* ServerSession)

Gibt das Schema der verwendeten XML zurück, wenn ein Nachverfolgungsprotokoll unter Verwendung der Methode *PostWebTrackLog(...)* bereitgestellt wird.

string LookupShortenedURL**(** *string* ServerSession, *string* ShortenedUrlKey, *bool* DisableLinkTracking)

Gibt die URL zurück, an die, basierend auf einem gekürzten URL-Schlüssel, weitergeleitet wird.

ShortenedUrlKey ist der gekürzte URL-Schlüssel.

DisableLinkTracking legt fest, ob die Verknüpfungsnachverfolgung deaktiviert werden soll.

void **PostWebTrackLog**(*string* ServerSession, *string* WebTrackLogXML)

Sendet eine Reihe von Nachverfolgungsprotokollelementen zur Datenbank. Webnachverfolgungsprotokolle werden in der Datenbanktabelle *WEB_TRACK_LOG* gespeichert.

WebTrackLogXML ist das XML-Dokument, das die Elemente enthält. Dieses XML-Dokument wird durch das XML-Schema definiert, das von *GetWebTrackLogSchema(...)* zurückgegeben wird.

void **PostWebTrackLogItem**(*string* ServerSession, *string LogType, datetime Tmestamp, int64* CustomerMessageID, *string* URL, *string* LinkName, *string* UserAgent, *string* BrowserType, *string* BrowserVersion, *string* PlatformName, *string* CustomValue1, *string* CustomValue2, *string* CustomValue3)

Sendet ein einzelnes Element des Nachverfolgungsprotokolls an die Datenbank. Webnachverfolgungsprotokolle werden in der Datenbanktabelle *WEB_TRACK_LOG* gespeichert.

LogType ist der Name des Protokolltyps.

Timestamp ist der Datums- und Zeitwert des Protokollelements.

CustomerMessageID bezieht sich auf einen Datensatz in der Datenbanktabelle CUSTOMER_MESSAGE. Diese ID identifiziert die dazugehörige Nachricht, wenn beispielsweise das Protokollelement auf einer E-Mail basiert, die an den Kunden gesendet wurde (E-Mail- und Verknüpfungsnachverfolgung).

URL ist die URL der Verknüpfung, die mit dem Protokollelement verbunden ist. Im Fall der Verknüpfungsnachverfolgung ist dies die URL, zu welcher der Kunde umgeleitet wurde.

LinkName ist der Name der Verknüpfung (oder URL), der bei der Verknüpfungsnachverfolgung verwendet wurde. Der Verknüpfungsname wird definiert, wenn eine Nachrichtenvorlage in Visual Dialogue entworfen wird.

UserAgent ist eine Zeichenfolge, die Informationen zur Anwendung enthält (z. B. Webbrowser oder E-Mail-Programm), die vom Kunden verwendet wurde (optional).

BrowserType ist der Name des Webbrowsers, der vom Kunden verwendet wurde (optional).

BrowserVersion ist die Version des Webbrowsers, der vom Kunden verwendet wurde (optional).

Platform ist eine Zeichenfolge, die das vom Kunden verwendete Betriebssystem anzeigt (optional).

„CustomValue1“, „CustomValue2“ und „CustomValue3“ sind benutzerdefinierte Zeichenfolgenwerte. Die angegebenen Werte werden in der Datenbanktabelle WEB_TRACK_LOG gespeichert. Die maximale Länge jedes Wertes beträgt 254 Zeichen.

string **ShortenAndNameURL**(*string* ServerSession, *string* OriginalUrl, *string* ShortUrlName, *bool* EnableLinkTracking, *bool* ReplaceExistingNamedUrl, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für `OriginalUrl` mit dem optionalen Präfix `ShortUrlPrefix` zurück, das an den benannten Teil des URL-Pfades angefügt wird, z. B. `http://shorturl.pb.com/Sonder-`

angebote/Weihnachten2013. Wenn EnableLinkTracking „true“ ist, werden Informationen zur Verknüpfungsnachverfolgung erfasst und der kurze URL-Name als Verknüpfungsname verwendet. Es wird bei jedem Abruf dieselbe URL zurückgegeben, es sei denn sie wurde vorher mit anderen Parametern abgerufen. In diesem Fall wird ein Fehler ausgegeben. Dieses Verhalten kann überschrieben werden, indem der Parameter ReplaceExistingNamedUrl auf „true“ gesetzt wird. Allerdings werden dann die zuvor generierten gekürzten URLs zur letzten OriginalUrl weitergeleitet. NB! Die Datenbanktabelle der gekürzten URLs ist für alle Kundendomänen der Instanz dieselbe. Eine benannte gekürzte URL darf nur für eine Domäne verwendet werden.

string **ShortenAndTrackURL**(*string* ServerSession, *string* OriginalUrl, *string* LinkName, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für OriginalUrl mit dem optionalen Präfix ShortUrlPrefix zurück, das an den codierten Teil der URL angefügt wird, z. B. http://shorturl.pb.com/Sommerkampagne/AhF56yx. Portrait Dialogue erfasst Informationen zur Verknüpfungsnachverfolgung für den angegeben LinkName. Es wird bei jedem Abruf eine andere URL zurückgegeben.

string **ShortenURL**(*string* ServerSession, *string* OriginalUrl, *string* ShortUrlPrefix)

Gibt eine gekürzte URL für OriginalUrl mit dem optionalen Präfix ShortUrlPrefix zurück, das an den codierten Teil der URL angefügt wird, z. B. http://shorturl.pb.com/Sommerkampagne/AhF56yx. Es wird bei jedem Abruf eine andere URL zurückgegeben.

void **UpdateWebPublicFile**(*string* ServerSession, *string* ContentID, *variant* ContentBinary, *string* Filename, *string* Description)

Aktualisiert den Inhalt und andere Eigenschaften einer veröffentlichten Datei. *ContentID* bleibt dabei erhalten.

AddWebPublicFile(...) gibt eine *Inhalts-ID* zurück, die automatisch der zu veröffentlichenden Datei zugewiesen wird. Eine Inhalts-ID ist eine Zeichenfolge, welche die Datei eindeutig identifiziert (GUID).

ContentID ist eine Zeichenfolge (GUID), welche die Datei eindeutig identifiziert.

ContentBinary ist der Inhalt der Datei, binär dargestellt oder als Byte-Array.

Filename ist der Name der Datei und der Erweiterungsteile. Pfadinformationen sind davon ausgeschlossen. *Beispiel:* meinlogo.jpg.

Description ist eine optionale Dateibeschreibung, die in Dialogue Database mit der Datei gespeichert wird.

Beispiele

Es stehen keine Beispiele zur Verfügung.

Kapitel 12

HQ Administration

In diesem Abschnitt:

- Portrait Shared Server 314
- Konfigurieren von Portrait HQ 314
- Konfigurieren von Portrait Shared Server 315
- Aktivieren der Kampagnengenehmigung 321
- Ändern der HQ-Benutzerberechtigungen 321
- Marketingaktivitäten konfigurieren 323
- Konfigurieren von Kundenkarten 327
- Ergebnisdaten-Integration 330
- Protokollierung von Daten aus externen Tools 342
- HQ-Fehlerbehebung 342

Portrait Shared Server

Portrait Shared Server besteht aus fünf Hauptkomponenten:

- **Portrait Shared Services (PSS)**: Stellt die Webdienste bereit, über die Portrait Suite-Anwendungen (Portrait Explorer, Miner, Dialogue und Interaction Optimizer) miteinander kommunizieren können.
- **Portrait Shared Repository (PSR)**: Stellt das Portrait Shared Repository und die Portrait Data Warehouse-Datenbanken bereit.
- **SharePoint Tasks (optional)**: Stellt eine integrierte Lösung zur Aufgabenverwaltung bereit, mit der Aufgaben, die Einzelpersonen in Portrait HQ zugewiesen sind, zur besseren Ansicht in SharePoint angezeigt werden. **Hinweis:** SharePoint Tasks ist für Portrait Dialogue und Portrait Interaction Optimizer optional und für Portrait Explorer nicht erforderlich.
- **Portrait Reports (optional)**: Stellt eine Sammlung vorgefertigter Berichte zur operativen Leistung von Portrait Interaction Optimizer und Portrait Dialogue bereit. **Hinweis:** Portrait Reports ist für Portrait Dialogue und Portrait Interaction Optimizer optional und für Portrait Explorer nicht erforderlich.
- **Portrait HQ**: Bietet ein zentrales Dashboard für Planung, Start und Überwachung von großen (1:1-)Marketingkampagnen. Neben Live-Zusammenfassungen Ihrer allgemeinen Marketingposition liefert Portrait HQ auch Echtzeitdaten, anhand derer Sie den Fortschritt von Kampagnen auswerten und, falls erforderlich, sofort Maßnahmen ergreifen können.

Konfigurieren von Portrait HQ

Konfigurieren der Bildwiederholrate

Bearbeiten Sie zur Neukonfiguration der maximalen Bildwiederholrate, die in der Portrait HQ-Client-Anwendung verwendet wird, die Datei `Web.config` auf der Portrait HQ-Website, um die Eigenschaft `MaxFrameRate` entsprechend zu definieren. Eine niedrigere Bildwiederholrate wird weniger Last auf den Client-Computer legen, jedoch in ‚hakeliger' Animation resultieren.

Konfigurieren der Portrait HQ-Anwendungsprotokollierung

Die Portrait HQ-Anwendungsprotokollierung kann über die Eigenschaften `LogTargets` und `LogLevel` in der Datei `Web.Config` konfiguriert werden. Diese Datei befindet sich auf dem Portrait Shared Server unter: `C:\Program Files (x86)\PST\Portrait Shared Server\Marketing HQ`.

Bearbeiten Sie zur Konfiguration der Portrait HQ-Protokollierungsziele entsprechend manuell die Eigenschaft `LogTargets` in der Datei `Web.config`. **Hinweis:** Momentan ist der einzige zulässige Wert `ClientFile`, der die Protokollierung an den isolierten Speicher der Silverlight-Anwendung weiterleitet.

Bearbeiten Sie zur Konfiguration der Protokollierungsstufe manuell die Eigenschaft *LogLevel* in der Datei `Web.config`, und wählen Sie eines der Folgenden aus:

- Kritisch
- Fehler
- Warnung

- Informationen
- Wortreich

Konfigurieren der Portrait Shared Repository-Datenbank

Bearbeiten Sie zur Neukonfiguration der Portrait Shared Services-Website, um auf eine andere *Portrait Shared Repository*-Datenbank zu verweisen, manuell die Datei `Web.config` auf der Portrait Shared Server-Website, um den „Verbindungsstring“ in der Eigenschaft `PsrEntities` korrekt im Abschnitt `ConnectionStrings` zu definieren. Die Website ist standardmäßig installiert in `C:\Program Files\PST\Portrait Shared Server\Portrait Shared Services`.

Konfigurieren von SharePoint

Bearbeiten Sie zur Neukonfiguration der Portrait Shared Server-Website, um auf einen anderen SharePoint-Server oder eine andere SharePoint-Website zu verweisen, manuell die Datei `web.config` auf der Portrait Shared Server-Website. Es müssen zwei Endpunktdefinitionen geändert werden – eine für die *Listen*-Dienste und eine für die *UserGroup*. Die Endpunkte enthalten jeweils eine URL zum entsprechenden Dienst auf der SharePoint Website. Die URLs sind in der Form: `http://servername/sitename/_vti_bin/servicename.asmx`, wobei

- *servicename* entweder Lists oder UserGroup ist
- *servername* ist der Name des Servers, auf dem SharePoint läuft, mit einer optionalen Portnummer für die SharePoint-Website. Zum Beispiel `myserver` oder `myserver:1234`.
- *sitename* ist der Name der Website innerhalb des SharePoint-Servers, welche die Listen und andere Artefakte beinhaltet, die von Portrait Shared Server benötigt werden. Standardmäßig wird die Website „Portrait“ genannt, was jedoch geändert werden kann, wenn erforderlich.

Konfigurieren von Portrait Shared Server

Konfigurieren der Windows-Authentifizierung

Interaction Optimizer und Portrait Explorer

Bearbeiten Sie zur Neukonfiguration von Portrait Shared Server, um die Windows-Authentifizierung zu aktivieren oder deaktivieren, die Datei `bin\config\Portrait.Mas.Cms.Services.config` auf der Portrait Shared Server-Website, um die Eigenschaft `enableWindowsAuthentication` entsprechend zu definieren. Die Website ist standardmäßig installiert in `C:\Program Files\PST\Portrait Shared Server\Portrait Shared Services`.

Portrait Dialogue

Selbst bei aktivierter Windows-Authentifizierung über die PSS-Installation können sich nur Benutzer mit Konten, die für die Verwendung der Windows-Authentifizierung in Portrait Dialogue konfiguriert wurden, auch ohne Benutzername und Kennwort authentifizieren.

Konfigurieren der Portrait Shared Server-Protokollierung

Portrait Shared Server schreibt Protokolleinträge in Dateien im Unterordner „LogFiles“ innerhalb des virtuellen Verzeichnisses, in dem die Website bereitgestellt wird. Standardmäßig erfolgt dies unter `C:\Program Files\PST\Portrait Shared Server\Portrait Shared Services`.

Sie können konfigurieren, was genau hier protokolliert werden soll, indem Sie die Datei `MH.Common.config` im Ordner *bin\config* auf der Portrait Shared Server-Website bearbeiten. Die Abschnitt `<loggingConfiguration>` enthält Verknüpfungsinformationen für Protokollierungsziele im Microsoft Enterprise Library „Protokollierungsblock“ XML-Format. Weitere Informationen über das Format und wie man es konfiguriert finden Sie in der Microsoft Dokumentation.

Standardmäßig wird eine Zusammenfassung von High-Level-Abrufen des Portrait Shared Server in der Datei „PortraitSharedServices.log“ protokolliert. Schwere Fehler werden auch im Windows-Ereignisprotokoll protokolliert.

Die Standard-Protokollierungskonfiguration in `MH.Common.config` definiert ein Set aus Protokollierungszielen und zeichnet Ereignisse für die Protokollierung (von einem festen Set von Protokollierungskategorien aufkommend) unter diesen Zielen auf.

Es werden folgende Protokollierungskategorien von Portrait Shared Server verwendet. Den Protokoll-Listener können folgende Kategorien im Abschnitt `<categorySources>` des Blocks `<loggingConfiguration>` in der Datei `MH.Common.config` zugewiesen werden.

- `PortraitSharedServices_Event` – Einträge, die zu den Windows-Ereignisprotokollen geleitet werden.
- `PortraitSharedServices_Log` – Elemente, die zur Datei `PortraitSharedServers.log` geleitet werden (Standardmäßig eingestellt, um nur Informationen der obersten Stufe, anstatt von detaillierten wortreichen Fehlerbehebungen zu protokollieren).
- `PortraitSharedServices_Analytics_Trace` – Nachverfolgen von ausgehenden Anrufen an Portrait Customer Analytics (Standardmäßig deaktiviert).
- `PortraitSharedServices_SharePoint_Trace` – Nachverfolgen von ausgehenden Aufrufen an SharePoint (Standardmäßig deaktiviert).
- `DataAccessLayerTracingFlatFile` – Nachverfolgen von ausgehenden Aufrufen an Portrait Kampagnenmanager (Standardmäßig deaktiviert).
- `ExceptionsFlatFile` – Interne Ausnahmen, die zur Datei `Exceptions.log` geleitet werden, normalerweise für Fehler, die innerhalb des Protokollierungsrahmens auftreten.

Standardmäßig werden nur Fehler protokolliert. Stellen Sie zur detaillierteren Protokollierung das `switchValue`-Attribut für die entsprechenden `<categorySources>` auf:

- **All** – alles protokollieren
- **Off** – Protokollierung deaktivieren
- **Critical** – nur kritische Ereignisse protokollieren
- **Error** – Fehler und kritische Ereignisse protokollieren
- **Information** – Informationen, Fehler und kritische Ereignisse protokollieren

Hinweis: Wenn Sie Konfigurationsänderungen in `MH.Common.config` vornehmen, werden diese Änderungen für Benutzer von Portrait HQ erst wirksam, wenn sie eine neue Sitzung starten (durch Schließen und erneutes Starten des Webbrowsers).

Wenn Sie die Datei `MH.Common.config` falsch konfigurieren kann die Protokollierung unspezifisch sein, und es wird schwierig zu diagnostizieren was falsch läuft. Wir empfehlen, dass Sie eine Sicherheits-

kopie der Datei machen bevor Sie Änderungen vornehmen, damit Sie zur funktionierenden Version zurückkehren können, falls Änderungen zu einem zerstörten System führen.

Konfigurieren der Kampagnen-Berichte

Übersicht

Auf den Kampagnenüberwachungsseiten in Portrait HQ stehen einige „Berichte“-Menüs zur Verfügung. Das System kann so konfiguriert werden, dass diese Menüs Links zu externen, webbasierten Berichtssystemen zur Verfügung stellen. Tatsächlich ist dieses System über URLs konfigurierbar, so dass diese Links auf jede URL im Kontext von jedem „Berichte“-Menü parametrisiert werden können.

Datenbankrelationen:

Die Konfiguration erfolgt durch das Hinzufügen von Einträgen in den beiden Relationen der „PortraitPSR“-Datenbank (Standardname) in der Portrait Shared Server-Installation. Diese zwei Relationen heißen „ExternalLink“ und „ExternalLinkParameter“ und stehen in Verbindung mit den Werten einer dritten Relation namens „ExternerLinkType“. Es gibt vier Verknüpfungstypen:

- **Kampagnen-Bericht:** Verknüpfungen dieser Art werden im Berichtsmenü im Abschnitt „Überwachung der Kampagne Key-Performance-Seite“ dargestellt. Diese Verknüpfungen können von der Kampagnen-Id parametrisiert werden.
- **Kampagnen-Aktivitätenbericht:** Verknüpfungen dieser Art werden im Berichtsmenü im Bereich „Marketing-Aktivitäten Performance“ der Seite „Kampagne Key-Performance“, sowie auch im Berichtsmenü im Abschnitt „Überwachung der Aktivität Key-Performance-Seite“ dargestellt. Diese Verknüpfungen können über die Kampagnen-Id und die Kampagnen-Aktivitäten-Id parametrisiert werden.
- **Angebotsbericht:** Verknüpfungen dieser Art sind im Berichtsmenü im Abschnitt „Angebot Performance“ der Seite „Kampagne Key-Performance“ und auf der Seite „Aktivitäten-Key-Performance“ dargestellt. Diese Verknüpfungen können über die Kampagnen-Id, die Kampagnen-Aktivitäten-ID und die Angebots-Id parametrisiert werden.
- **Kanal-Bericht:** Verknüpfungen dieser Art werden im Berichtsmenü im Abschnitt „Kanal-Performance“ und der Seite „Aktivitäten-Key-Performance“ dargestellt. Diese Verknüpfungen können über die Kampagnen-Id, die Kampagnen-Aktivitäten-ID, die Behandlungs-Id und die Kanal-ID parametrisiert werden.

Um dem System eine Verknüpfung hinzuzufügen, fügen Sie in der Relation „ExternalLink“ einen Datensatz mit einem Namen für die Verknüpfung, einer Basis-URL, der ExternalLinkTypeId (siehe Relation „ExternalLinkType“ hinzu, um die IDs zu jedem oben genannten Verknüpfungstypen zu finden) und alle anderen benötigten Felder. Beachten Sie, dass das ID-Feld automatisch generiert wird und dass einige der Felder derzeit nicht in der Konfiguration verwendet werden, aber für administrative Zwecke (z. B. Version) nützlich sind.

Um Linkparameter einzustellen, fügen Sie eine oder mehrere Zeilen zur Relation „ExternalLinkParameter“ mit Angabe der ExternalLinkId (die ID der oben genannten externen Verknüpfung) und des Parameternamens, der in der URL integriert wird, welche generiert wird, wenn die Verknüpfung angeklickt wurde und ein „Mapping“ eines der folgenden Dinge tut:

- **CampaignId**: um den Wert für die Kampagnen-ID zu übergeben
- **ActivityId**: um den Wert für die Kampagnen-Aktivitäten-ID zu übergeben

- **OfferId**: um den Wert für die Angebots-ID zu übergeben
- **TreatmentId**: um den Wert für die Verfahrens-ID zu übergeben
- **ChannelId**: um den Wert für die Kanal-ID zu übergeben

Zum Beispiel, um einen Link mit dem Namen „Die Kennzahlen des Monats“ einzustellen, der mit einem Berichtssystem verbunden wird, hat die URL folgende Form:

```
http://myreportingsystem?ActivityParameter=22&CampaignParameter=1003
```

Fügen Sie für Kampagnenaktivitäten der Relation „ExternerLink“ eine Zeile mit dem Name=Die Kennzahlen des Monats, Url=http://myreportingsystem, ExternerLinkTypId=2, hinzu (dies ergibt ID = 5). Fügen Sie danach eine Zeile zur Relation „ExternalLinkParameter“ mit den Werten ExterneLinkId=5, Name=ActivityParameter und Mapping=ActivityId hinzu. Fügen Sie zum Schluss noch eine andere Zeile zur Relation „ExternalLinkParameter“ mit den Werten ExterneLinkId =5, Name=CampaignParameter und Mapping=CampaignId hinzu.

Beachten Sie, dass Sie die Basis-URL parametrisieren können (aber nicht müssen). Zum Beispiel: Wenn die oben genannte URL auf http://myreportingsystem?myparametername=myparametervalue gesetzt wurde, wird dieselbe Konfiguration in diesem Link verwendet:

```
http://myreportingsystem?myparametername=myparametervalue&ActivityParame-
ter=22&CampaignParameter=1003
```

Konfigurieren der Quicklinks in MyView

Änderungen im Abschnitt „Quicklinks“ auf der Seite „MyView“ werden in der Relation `ApplicationLink` der PSR-Datenbank ausgeführt. Die Relation enthält ein paar Standardverknüpfungen, die durch Einstellen des Feldes „Aktiviert“ auf den Wert „0“ deaktiviert werden können. Hinweis: Ändern Sie keine Verknüpfungen mit einem `SystemName`; diese werden zwingend benötigt. Neue Verknüpfungen können durch das Hinzufügen neuer Zeilen in der Relation `ApplicationLink` hinzugefügt werden. Die folgende Tabelle zeigt die benötigten Felder.

Spaltenname	Beschreibung
Name	Der Text der am Link angezeigt wird.
Uri	Ein neues Browserfenster wird geöffnet, wenn ein Benutzer auf die Verknüpfung klickt.
Enabled	Aktivieren/deaktivieren der Verknüpfung. Setzen die den Wert auf „1“ um zu aktivieren.

NB! Der Inhalt der Relation `ApplivationsLink` wird nicht exakt zu den Quicklinks in LiveView passen. Der Grund dafür ist, dass:

- Es gibt zwei Verknüpfungen, die zu den Quicklinks hinzugefügt wurden, aber nicht in der Relation `ApplicationLink` vorkommen: **Neue Kampagne erstellen** und **Neue Aufgabe erstellen**.
- In der Relation gibt es eine Verknüpfung zur Anwendung „Visual Dialog“. Diese Verknüpfung ist unter „Tools“ der oberen Menüleiste verfügbar.

Aktivieren von SSL/HTTPS

Führen Sie zur Aktivierung von SSL/HTTPS für Portrait HQ und Portrait Shared Server folgende Schritte aus:

1. Aktualisieren Sie die Konfiguration für die Website, die Portrait HQ und Portrait Shared Server hostet:
   a. Erwerben Sie das notwendige SSL-Zertifikat, und verwenden Sie dieses Zertifikat für die HTTP-Site-Bindung.
   b. Entfernen Sie die HTTP-Site-Bindung.
   c. Aktualisieren Sie die SSL-Einstellungen für Portrait HQ und Portrait Shared Server, um SSL erforderlich zu machen, aber Kundenzertifikate zu ignorieren.
2. Aktualisieren Sie die Konfigurationsdatei `web.config` für Portrait Shared Server:
   a. Suchen Sie die Portrait Shared Server`web.config`-Datei unter: `\<Installationsverzeichnis>\PST\Portrait Shared Server\Portrait Shared Services\Web.config`.
   b. Setzen Sie den Sicherheitsmodus in der Bindung `webHttpBinding` auf „Transport“.
      ```
      <security mode="transport">
      ```
   c. Stellen Sie sicher, dass der HTTP-Zugriff für die Dienstmetadaten deaktiviert ist.
      ```
      <serviceMetadata httpGetEnabled="False" />
      ```
   d. Entfernen Sie die Kommentare für alle Abschnitte innerhalb des folgenden Satzes:
      ```
      <!-- Uncomment this section to enable SSL access. -->
      ```
   e. Fügen Sie Kommentare für alle Abschnitte innerhalb des folgenden Satzes hinzu:
      ```
      <!-- Comment this section to enable SSL access. -->
      ```
   f. Speichern und schließen Sie die Datei.
3. Aktualisieren Sie die Konfigurationsdatei `web.config` für Portrait HQ:
   a. Suchen Sie die Datei `web.config` von Portrait HQ unter: `\Installationsverzeichnis\PST\Portrait Shared Server\Marketing HQ\Web.config`.
   b. Entfernen Sie die Kommentare für alle Abschnitte innerhalb des folgenden Satzes:
      ```
      <!-- Uncomment this section to enable SSL access. -->
      ```
   c. Fügen Sie Kommentare für alle Abschnitte innerhalb des folgenden Satzes hinzu:
      ```
      <!-- Comment this section to enable SSL access. -->
      ```
   d. Speichern und schließen Sie die Datei.
4. Aktualisieren Sie die Konfigurationsdatei `web.config` für `DecisionsWCFWebService`:
   a. Suchen Sie die Datei `web.config`, indem Sie das virtuelle Verzeichnis `DecisionsWCFWebService` in IIS Manager durchsuchen.
   b. Entfernen Sie die Kommentare für alle Abschnitte, vor denen folgender Ausdruck steht:
      ```
      <!-- Uncomment this section to enable SSL access. -->
      ```

c. Speichern und schließen Sie die Datei.

5. Aktualisieren Sie die Konfigurationsdatei `web.config` für `IOBridgeWCFWebService`:

   a. Suchen Sie die Datei `web.config`, indem Sie das virtuelle Verzeichnis `IOBridgeWCFWebService` in IIS Manager durchsuchen.
   b. Entfernen Sie die Kommentare für alle Abschnitte, vor denen folgender Ausdruck steht:

   ```
   <!-- Uncomment this section to enable SSL access. -->
   ```

   c. Speichern und schließen Sie die Datei.

6. Aktualisieren Sie die Konfigurationsdatei `web.config` für `IOWCFWebService`:

   a. Suchen Sie die Datei `web.config`, indem Sie das virtuelle Verzeichnis `IOWCFWebService` in IIS Manager durchsuchen.
   b. Entfernen Sie die Kommentare für alle Abschnitte, vor denen folgender Ausdruck steht:

   ```
   <!-- Uncomment this section to enable SSL access. -->
   ```

   c. Speichern und schließen Sie die Datei.

7. Aktualisieren Sie die Konfigurationsdatei `DatasourceDefExporter.exe.config` für das Tool „DatasourceDefExporter.exe“:

   a. Suchen Sie die Datei `DatasourceDefExporter.exe.config` unter: `\installation_directory\PST\\Portrait IO\Tools\`
   b. Entfernen Sie die Kommentare für alle Abschnitte innerhalb des folgenden Satzes:

   ```
   <!-- Uncomment this section to enable SSL access. -->
   ```

   c. Fügen Sie Kommentare für alle Abschnitte innerhalb des folgenden Satzes hinzu:

   ```
   <!-- Comment this section to enable SSL access. -->
   ```

   d. Speichern und schließen Sie die Datei.

8. Aktualisieren Sie die Konfigurationsdatei `DecisionTransfer.exe.config` für das Tool „DecisionTransfer.exe“:

   a. Suchen Sie die Datei `DecisionTransfer.exe.config` unter: `\installation_directory\PST\\Portrait IO\Tools\`
   b. Entfernen Sie die Kommentare für alle Abschnitte innerhalb des folgenden Satzes:

   ```
   <!-- Uncomment this section to enable SSL access. -->
   ```

   c. Fügen Sie Kommentare für alle Abschnitte innerhalb des folgenden Satzes hinzu:

   ```
   <!-- Comment this section to enable SSL access. -->
   ```

   d. Speichern und schließen Sie die Datei.

Aktivieren der Kampagnengenehmigung

Portrait HQ kann so konfiguriert werden, dass Kampagnenänderungen erst genehmigt werden müssen, bevor die Kampagne gestartet werden kann. Nur Portrait HQ-Benutzer mit der Berechtigung *Kampagne genehmigen* dürfen die Kampagne genehmigen.

So konfigurieren Sie eine Kampagnengenehmigung:

1. Öffnen Sie die Datei `web.config` im Ordner `<Installationsverzeichnis>\PST\Portrait Shared Server\Marketing HQ\`.
2. Legen Sie für den Parameter `CampaignApprovalEnabled` den Wert `True` fest.

Hinweis: Sobald dieser Parameter aktiviert wurde, müssen alle Kampagnenänderungen von HQ-Benutzern mit entsprechender Berechtigung genehmigt werden.

Ändern der HQ-Benutzerberechtigungen

Gehen Sie wie folgt vor, um die HQ-Benutzerberechtigungen zu ändern.

1. Melden Sie sich am Portrait HQ-Server an, und öffnen Sie die Datei `Portrait.Mas.Cms.Services.Authorization.config` unter: `/pst/Portrait Shared Server/Portrait Shared Services/Bin/config/`.
2. Aktualisieren Sie die Benutzerberechtigungen nach Ihren Wünschen.

   Note: Nachfolgend ist der Standardsatz an Berechtigungen für jede HQ-Rolle aufgeführt. Die Bedeutung jeder einzelnen Berechtigung wird in der nachfolgenden Tabelle erläutert.

```
<RolePermissions>
    <Role Name= "SeniorManagement">
        <Permission Name="AllowedToLogOn" />
        <Permission Name="AllowedToLogOnToEmarketingDesigner" />
        <Permission Name="DeleteTask" />
        <Permission Name= "EditTask" />
        <Permission Name= "ReadCampaign" />
        <Permission Name= "ReadOffer" />
        <Permission Name= "ReadTask" />
    </Role>
    <Role Name="BusinessStakeholder">
        <Permission Name="AllowedToLogOn" />
        <Permission Name="AllowedToLogOnToEmarketingDesigner" />
        <Permission Name="DeleteOffer" />
        <Permission Name="EditOffer" />
        <Permission Name="ReadCampaign" />
        <Permission Name="ReadOffer" />
        <Permission Name="ReadTask" />
    </Role>
        <Role Name="Creative">
            <Permission Name="AllowedToLogOn" />
            <Permission Name="AllowedToLogOnToEmarketingDesigner" />
```

```
            <Permission Name="EditEmarketingMailTemplates" />
            <Permission Name="EditTask" />
            <Permission Name="ReadCampaign" />
            <Permission Name="ReadOffer" />
            <Permission Name="ReadTask" />
        </Role>
        <Role Name="CustomerInsight">
            <Permission Name="AllowedToLogOn" />
            <Permission Name="AllowedToLogOnToEmarketingDesigner" />
            <Permission Name="EditTask" />
            <Permission Name="ReadCampaign" />
            <Permission Name="ReadOffer" />
        <Permission Name="ReadTask" />
        </Role>
        <Role Name="DirectMarketer">
            <Permission Name="AllowedToLogOn" />
            <Permission Name="AllowedToLogOnToEmarketingDesigner" />
            <Permission Name="EditEmarketingMailTemplates" />
            <Permission Name="AllowedToLogOnToVisualDialog" />
            <Permission Name="DeleteItem" />
            <Permission Name="DeleteOffer" />
            <Permission Name="DeleteTask" />
            <Permission Name="EditCampaign" />
            <Permission Name="EditOffer" />
            <Permission Name="EditTask" />
            <Permission Name="ReadCampaign" />
            <Permission Name="ReadOffer" />
            <Permission Name="ReadTask" />
            <Permission Name="EditGlobalSelections" />
        </Role>
        <Role Name="MarketingOperations">
            <Permission Name="AllowedToLogOn" />
            <Permission Name="AllowedToLogOnToEmarketingDesigner" />
            <Permission Name="EditEmarketingMailTemplates" />
            <Permission Name="AllowedToLogOnToVisualDialog" />
            <Permission Name="DeleteItem" />
            <Permission Name="DeleteOffer" />
            <Permission Name="DeleteTask" />
            <Permission Name="EditCampaign" />
            <Permission Name="EditOffer" />
            <Permission Name="EditTask" />
            <Permission Name="ReadCampaign" />
            <Permission Name="ReadOffer" />
            <Permission Name="ReadTask" />
            <Permission Name="EditGlobalSelections" />
            </Role>
```

Berechtigung	Beschreibung
AllowedToLogOn	Anmelden bei der Portrait HQ-Anwendung zulässig.
AllowedToLogOnToEmarketingDesigner	Anmelden bei der Message Designer. Nachrichtenvorlagen bestimmen den Inhalt, der bei Outbound-Kommunikationen gesendet wird.
EditEmarketingMailTemplates	Bearbeiten von Nachrichtenvorlagen im Message Designer zulässig.

Berechtigung	Beschreibung
AllowedToLogOnToVisualDialog	Starten von Visual Dialogue über eine Verknüpfung in Portrait HQ.
ReadCampaign	Anzeigen von Kampagnen, Selektionen, Nachrichten und Listen ohne deren Änderung.
EditCampaign	Ändern der Kampagnen, Selektionen, Nachrichten und Listen.
DeleteItem	Löschen der Kampagnen, Selektionen, Nachrichten und Listen.
ReadOffer	Anzeige der Angebote ohne deren Änderung.
EditOffer	Ändern der Angebote.
DeleteOffer	Löschen der Angebote.
ReadTask	Anzeigen der Aufgaben ohne deren Änderung.
EditTask	Ändern der Aufgaben.
DeleteTask	Löschen der Aufgaben.
EditGlobalSelections	Erstellen, Ändern und Entfernen der globalen Selektionen (globale Selektionen gelten für alle Kampagnen).

Marketingaktivitäten konfigurieren

Aktivitätstypen konfigurieren

Aktivitätstypen werden verwendet, um Kampagnenaktivitäten zu gruppieren. Portrait HQ kann konfiguriert werden, um Empfehlungen in Bezug auf ihren Aktivitätstyp einzustufen. Die Konfiguration der Liste der verfügbaren Aktivitätstypen erfolgt durch Änderung des Inhalts der Relation `CampaignActivityType` in der PSR-Datenbank. Die Relation `CampaignActivityType` enthält die folgenden Spalten:

Spaltenname	Beschreibung
Name	Eindeutiger Name der benutzt wird, um den Aktivitätentyp zu repräsentieren.
Description	Beschreibung für den Aktivitätentyp.
SupportsAnonymous	Auf 1 setzen, wenn dieser Aktivitätentyp auf unbekannte Kunden abzielt, ansonsten auf 0 setzen.
SupportsIdentified	Auf 1 setzen wenn dieser Aktivitätentyp auf bekannte Kunden abzielt, ansonsten auf 0 setzen.

Beispiel

Der Standardsatz von verfügbaren Aktivitätstypen wird durch Ausführung der folgenden SQL-Anweisungen für die PSR-Datenbank festgelegt:

```
INSERT INTO CampaignActivityType(Name, Description, SupportsAnonymous, Sup-
portsIdentified) VALUES('Acquisition', 'Acquisition.', 1, 1)
INSERT INTO CampaignActivityType(Name, Description, SupportsAnonymous, Sup-
portsIdentified) VALUES('Revenue', 'Revenue.', 0, 1)
INSERT INTO CampaignActivityType(Name, Description, SupportsAnonymous, Sup-
portsIdentified) VALUES('Retention', 'Retention.', 0, 1)
INSERT INTO CampaignActivityType(Name, Description, SupportsAnonymous, Sup-
portsIdentified) VALUES('Service', 'Service.', 1, 1)
INSERT INTO CampaignActivityType(Name, Description, SupportsAnonymous, Sup-
portsIdentified) VALUES('Loyalty', 'Loyalty.', 0, 1)
```

Konfigurieren von Aktivitätsuntertypen

Marketingaktivitäten können optional Aktivitätsuntertypen zugewiesen werden. Sie werden verwendet, um Kampagnenaktivitäten zusätzlich zu den Aktivitätstypen zu gruppieren. Portrait HQ kann konfiguriert werden, um Empfehlungen in Bezug auf den Aktivitätsuntertyp einzustufen. Die Konfiguration der Liste der verfügbaren Aktivitätsuntertypen erfolgt durch Änderung des Inhalts der Relation `CampaignActivitySubType` in der PSR-Datenbank. Die Relation `CampaignActivitySubType` enthält die folgenden Spalten:

Spaltenname	Beschreibung
Name	Eindeutiger Name, der verwendet wird, um den Aktivitätenuntertypen zu repräsentieren.
Description	Beschreibung für den Aktivitätenuntertypen.

Beispiel

Der Standardsatz von verfügbaren Aktivitätsuntertypen wird durch Ausführung der folgenden SQL-Anweisungen für die PSR-Datenbank festgelegt:

```
INSERT INTO CampaignActivitySubType(Name, Description) VALUES('Loan', 'Loan.')
INSERT INTO CampaignActivitySubType(Name, Description) VALUES('Mortgage',
'Mortgage.')
INSERT INTO CampaignActivitySubType(Name, Description) VALUES('Current Ac-
count', 'Current Account.')
```

Konfigurieren von Aktivitäteneigenschaften

Marketingaktivitäten können optional mit einer Priorität versehen werden. Diese Priorität kann benutzt werden, um Marketingaktivitäten einzuordnen, wenn sie auf eingehenden Kanälen präsentiert werden. Die Konfiguration der Liste der verfügbaren Marketingaktivitätseigenschaften erfolgt durch Änderung des Inhalts der Relation `PriorityType` in der PSR-Datenbank. Die Relation `PriorityType` enthält die folgenden Spalten:

Spaltenname	Beschreibung
Name	Eindeutiger Name der einen Prioritätentyp repräsentiert.

Spaltenname	Beschreibung
Description	Beschreibung für den Prioritätentyp.
PriorityValue	Wert für den Prioritätentyp, welcher genutzt wird, um die Priorität für eine Aktivität zu repräsentieren Dieser Wert wird benutzt, um Aktivitäten einzuordnen. Ein höherer Wert resultiert in einem höheren Rang.

Beispiel

Der Standardsatz von verfügbaren Prioritäten wird durch Ausführung der folgenden SQL-Anweisungen für die Portrait Shared Repository festgelegt:

```
INSERT INTO PriorityType(Name, Description, PriorityValue) VALUES('Highest',
 'Highest.', 10000)
INSERT INTO PriorityType(Name, Description, PriorityValue) VALUES('High',
'High.', 5000)
INSERT INTO PriorityType(Name, Description, PriorityValue) VALUES('Medium',
 'Medium.', 1000)
INSERT INTO PriorityType(Name, Description, PriorityValue) VALUES('Low',
'Low.', 500)
INSERT INTO PriorityType(Name, Description, PriorityValue) VALUES('Lowest',
 'Lowest.', 100)
```

Konfigurieren von Kanälen

Konfigurieren von Kanälen

In Portrait HQ wählen Benutzer durch Klicken auf Umschaltflächen auf der Seite „Marketingaktivitäten“, welche Inbound- und Outbound-Kanäle eine Marketingaktivität verwenden soll.

Zum Hinzufügen, Entfernen oder Ändern eines Kanals müssen Sie Änderungen in den Relationen „TreatmentType“ und „ChannelType“ im Portrait Shared Repository vornehmen.

Tabelle 1: ChannelType-Relation

Spaltenname	Beschreibung
Name	Der Kanalname.
Description	Beschreibung für den Kanal.
PcmChannelName	Der Name des Kanals in Portrait Dialogue. Bei neuen Kanälen sollte dieses Feld normalerweise leer bleiben.

Tabelle 2: TreatmentType-Relation

Spaltenname	Beschreibung
Name	Der Name des Behandlungstyps. Muss eindeutig sein.

Spaltenname	Beschreibung
ShortName	Die Kurzform des Namen des Behandlungstyps. Dies ist der Text, der unter dem Kanalsymbol auf der Seite „Marketingaktivitäten“ angezeigt wird.
Description	Beschreibung für den Behandlungstyp.
ChannelTypeId	Die Kanaltyp-ID. Muss eine ID von einem der Kanäle in der ChannelType-Relation sein.
SupportsAnonymous	Falls anonyme Behandlung unterstützt wird. Setzen Sie diesen Wert auf 1 für Behandlungstypen, bei denen der behandelte Kunde unbekannt ist.
SupportsIdentified	Falls die identifizierte Behandlung unterstützt wird. Setzen Sie diesen Wert auf 1 für Behandlungstypen, bei denen der behandelte Kunde bekannt ist.
IsControlGroup	Falls behandelte Kunden als Kontrollgruppe behandelt werden sollen. Sollte normalerweise 0 sein.
SymbolData	Das Symbol des Behandlungstyps wird als XAML-Grafikpfad angezeigt (Vektorgrafiken).
DefaultTreatmentCost	Optionaler Wert zu den Standard-Behandlungskosten.
PcmDefaultBranchType	Falls der Behandlungstyp mit einer Portrait Dialogue-Verzweigung verknüpft ist. Bei neuen Kanälen sollte dieses Feld normalerweise leer bleiben.
PcmDefaultOperationType	Falls der Behandlungstyp einen Standard-Vorgangstyp in Portrait Dialogue besitzt. Bei neuen Kanälen sollte dieses Feld normalerweise leer bleiben.
IsOutbound	Falls der Behandlungstyp eine Inbound- oder Outbound-Behandlung angibt. Wird auf der Seite „Marketingaktivitäten“ unter „Zu verwendende Outbound-Kanäle“ angezeigt, wenn der Wert auf 1 gesetzt wird bzw. unter „Zu verwendende Inbound-Kanäle“, wenn der Wert auf 0 gesetzt wird. Outbound-Behandlungen erfolgen in der Regel mithilfe von Portrait Dialogue, wohingegen Inbound-Behandlungen in der Regel mithilfe von Interaction Optimizer erfolgen.

Hinzufügen eines neuen Kanals zur Seite „Marketingaktivitäten“

Wenn Sie einen komplett neuen Kanal auf der Seite „Marketingaktivitäten“ hinzufügen möchten, führen Sie folgende Schritte aus:

1. Fügen Sie einen neuen Datensatz zur Relation `ChannelType` hinzu.

2. Fügen Sie einen neuen Datensatz zur Relation `TreatmentType` hinzu, und legen Sie im Feld `ChannelTypeId` fest, dass der Datensatz, den Sie gerade erstellt haben, in der Relation `Channel-Type` verknüpft wird.

Ändern des Texts des Kanalsymbols auf der Seite „Marketingaktivität“

Wenn Sie den Text, der auf der Seite Marketingaktivitäten unter den Kanalsymbol angezeigt wird, ändern möchten, führen Sie folgenden Schritte aus:

1. Finden eines relevanten Datensatzes in der Relation „TreatmentType“.
2. Ändern Sie den Wert im Feld „ShortName“.

Entfernen eines Kanals, welcher auf der Seite Marketingaktivitäten angezeigt wird

Warnung: Das Entfernen einen Kanals von der Seite Marketingaktivitäten setzt ein Entfernen des Behandlungstyp von der Datenbank voraus. Dies kann schwerwiegende Folgen für laufende Kampagnen haben und sollte im Idealfall nur auf einem neuen System ohne Kampagnen durchgeführt werden (oder zumindest ein System ohne laufende Kampagnen).

1. Suchen Sie den relevanten Datensatz in der Relation „TreatmentType“ und notieren Sie die entsprechende ID.
2. Löschen Sie sämtliche Datensätze in der Behandlungsrelation, die im Bezug zum Behandlungstyp stehen, den Sie einfach durch einen Blick in die Spalte `TreatmentTypeId` finden und mit der entsprechenden ID des Verfahrenstyps verbinden, den Sie löschen möchten. **Hinweis:** Falls einer der Datensätze, den Sie in der Behandlungsrelation löschen möchten, in Bezug zu einem Datensatz aus der Relation `TreatmentOfferForecast` steht, müssen Sie diesen Datensatz zuerst löschen. Sie können ihn anhand der Behandlungs-ID finden, wenn eine Behandlung eine Angebotsvorschau besitzt. Danach prüfen Sie einfach, ob Sie in der Relation `TreatmentOfferForecast` einen passenden Wert in der Spalte `TreatmentId` finden.
3. Löschen des Behandlungstypen.

Konfigurieren von Kundenkarten

Kundenkarten werden in Portrait HQ angezeigt, wenn Sie bei der Anzeige oder Bearbeitung einer Auswahl in einem unterstützten Dialog im Design-Schritt der Kampagne auf die Schaltfläche „Beispiel anzeigen“ klicken. In der daraufhin geöffneten Ansicht wird eine Reihe von Karten angezeigt. Dieser Abschnitt beschäftigt sich mit dem Konfigurieren des Inhalts dieser Karten und beschreibt sowohl das Erscheinungsbild als auch die Daten.

Übersicht

Die Konfiguration der Karten stützt sich auf die folgenden Komponenten im System:

- Die Kartenvorlage **Card.xaml**, die hier als „die Karte“ bezeichnet wird. Sie finden sie im Verzeichnis „ClientBin\Resources“ des Webservers, auf dem sich die Portrait HQ-Anwendung befindet. Dasselbe Verzeichnis enthält auch Ressourcen, die von der Karte genutzt werden.

- Ein oder mehrere benutzerdefinierte SQL-Skripte mit dem Namen **PortraitHQ_CustCards_<domainId>**. Sie werden im Portrait HQ-Tool erzeugt und konfiguriert und befinden sich im Ordner „Resources/SQL Repository/Customized SQLs“ der Datenbankinstanz, für deren Nutzung Dialogue Admin konfiguriert wurde. Dies wird hier als „das SQL-Skript“ bezeichnet.
- Der Webdienst, den der Portrait HQ-Client aufruft, um Kundenkartendaten abzurufen. Dieser wird hier als „der Webdienst“ bezeichnet.
- Eine Konvertierungsressource, **DictionaryStringToValueConverter**, die innerhalb des Karten-XAML genutzt wird, um die vom Webdienst abgerufenen Daten in Werte umzuwandeln, die in der Karte verwendet und angezeigt werden können. Diese wird hier als „der Konverter“ bezeichnet.

Die Kartenvorlage

Bei der Kartenvorlage handelt es sich um eine standardmäßige XAML-Datei, die ein Markup enthält, das sowohl das Aussehen der Karte als auch die Daten enthält, mit denen die visuellen Elemente verbunden sind. Das XAML kann entweder manuell oder in einem geeigneten Editor, normalerweise MS Expression Blend, bearbeitet werden.

Zur Laufzeit ist der Daten-Kontext für die Karte ein Objekt, das eine Eigenschaft „CardData“ enthält und die Quelle für die in der Karte verwendeten Daten bildet. Auf die Daten wird über „Datenbindung“ zugegriffen. Das ist ein Standardmechanismus in XAML und Silverlight.

Die Eigenschaft „CardData“ ist eigentlich ein „Schlüsselwert“-Verzeichnis, in dem sowohl Schlüssel als auch Werte Strings sind. Mithilfe des DictionaryStringToValueConverter wird auf die Werte in CardData zugegriffen, und sie werden in geeignete Datentypen oder formatierte Strings umgewandelt. Die Schlüssel in CardData entsprechen Spaltenamen in der durch das SQL-Skript erzeugten Tabelle.

Es gibt sehr wenige Einschränkungen hinsichtlich dessen, was innerhalb der Karte gemacht werden kann. Das bedeutet, dass es „nur“ Standard-XAML ist und alles möglich ist, was in einem standardmäßigen Silverlight-Steuerelement möglich ist. Beachten Sie jedoch:

- Vergewissern Sie sich, eine Datensicherung der mitgelieferten Beispiele erzeugt zu haben, bevor Sie bearbeitete Versionen erstellen!
- Die Größe (Breiten- und Höhenattribute des Hauptelements) der Karte dürfen nicht verändert werden, weil sie in eine Komponente mit feststehender Größe innerhalb der Anwendung passen muss. Beachten Sie jedoch, dass Karten in der Anwendung vergrößert bzw. verkleinert werden können, sodass es möglich ist, den Inhalt der Karte mehr oder weniger detailliert zu gestalten als im Beispiel.
- Es sollte jedoch darauf geachtet werden, die Karte so zu gestalten, dass das Rendern nicht zu aufwendig wird. Zum Beispiel: Bei der mit Portrait HQ gelieferten Beispielkarte wird der Schatten auf dem Rand auf einen Dummy-Rand statt auf den gesamten Karteninhalt angewandt. Effekte, wie Schatten bei vielen Elementen oder die Aufnahme komplexer Vektorgrafiken, können sich auf die Performance bei der Anzeige auswirken.

Das SQL-Skript

Die Daten, die auf einer Kundenkarte angezeigt werden stammen aus einem Kundendatensatz in einer Portrait Dialogue-Domäne. Die herangezogene Domäne wird durch den unterstützen Dialog bestimmt, der erstellt oder bearbeitet wird. Für jede bei Auswahlen zu verwendende Domäne muss daher ein SQL-Skript eingerichtet sein, um Kundendaten für Karten bereitzustellen. Die Skriptnamen müssen das Format **PortraitHQ_CustCards_<domainId>** haben, z. B. 'PortraitHQ_CustCards_1002'.

Die Skripte übernehmen einen Parameter „cust_ids“, bei dem es sich um eine Liste von Kunden-IDs handelt, die eine Gruppe von Datensätzen mit jeweils einer Zeile pro Kunden-ID zurückgibt. Die Spaltennamen des zurückgegebenen Datasets bilden schließlich die Schlüssel im CardData-Objekt, mit dem die Karte verbunden ist, und die Werte in der Tabelle bilden die Werte in CardData.

Beachten Sie, dass der Parameter „cust_ids“, der an das Skript übergeben wird, eine durch Kommas getrennte Liste von nicht in Anführungszeichen stehenden Integer-Werten ist (d. h. „1001,1002,2022“) und ist bei der Verwendung dieser Werte zur Auswahl von Datensätzen zu berücksichtigen, wenn das Feld „cust_id“ nicht numerisch ist. Im Beispiel-Skript werden Datensätze folgendermaßen ausgewählte: „cust_id in ({STRINGPARAM cust_ids})“, was auf „cust_id in (1001,1002,2022)“ erweitert wird, was fehlschlägt, wenn ein fehlerhafter Datensatz vorhanden ist, bei dem cust_id nicht in ein Integer umgewandelt werden kann. Eine Lösung wäre: "((ISNUMERIC(cust_id)=1) AND CAST(cust_id as int) in ({STRINGPARAM cust_ids}))".

Beachten Sie auch, dass die Vorverarbeitung dieser SQL-Skripte durch Kommentare gestört werden kann. Kommentare sind eine häufige Ursache für unerklärliche Fehler.

Die von den SQL-Skripten zurückgegebenen Werte können einen beliebigen Typ aufweisen. Beachten Sie jedoch, dass alle Werte in Strings umgewandelt werden, wenn sie vom Webdienst an den Browser übergeben werden. Diese Strings werden auf dem PSS-Server erstellt, jedoch immer in einer Weise, die unabhängig von den regionalen Einstellungen ist. Dies wird vom DictionaryStringToValueConverter berücksichtigt.

DictionaryStringToValueConverter

Der Konverter hat zwei Aufgaben. Zunächst ermöglicht er den Zugriff auf die Werte in CardData über die bekannten Schlüssel (die Spaltennamen aus dem SQL-Skript). Zweitens ermöglicht er die Umformung dieser Werte in bestimmte Typen oder eine bestimmte Formatierung.

Dem Konverter muss immer ein Parameter übergeben werden, bei dem es sich tatsächlich um eine Liste durch „|“ getrennter Argumente handelt. In seiner einfachsten Form kann dieser Parameter jedoch einfach nur der Name des Schlüssels sein. In diesem Fall ist das Ergebnis der entsprechende Stringwert aus CardData.

Der Parameter für den Konverter kann komplexer gestaltet sein und die Form „<KeyName>|<OutputType>|<Modifier>“ haben. Die folgenden Ausgabetypen werden unterstützt:

- **Boolesch**: Der Stringwert wird als Boolesch interpretiert. Eine leerer String, der Groß- und Kleinschreibung nicht berücksichtigende Wert „false“ und Strings, die als die Zahl Null ausgewertet werden können, geben immer „false“ zurück, alles andere ist „true“. Es wird kein Modifizierer benötigt.
- **Sichtbarkeit**: Der Stringwert wird, wie oben, als Boolesch interpretiert. Jedoch ist der Rückgabewert Sichtbarkeit.Sichtbar oder Sichtbarkeit.Reduziert statt „true“ bzw. „false“. Es wird kein Modifizierer benötigt.
- **Zahl**: Der Stringwert wird als Zahl interpretiert und das Zahlenobjekt wird zurückgegeben. Ein leerer String wird als Null interpretiert. Es wird kein Modifizierer benötigt.
- **FormatierterString**: Der Stringwert wird in einem .NET-Framework-Stringausdruck, wie im Modifizierer angegeben, verwendet. Eine gute Ressource hierfür finden Sie unter: http://blog.stevex.net/string-formatting-in-csharp/
- **FormatierteZahl**: Der Stringwert wird als Zahl interpretiert und dann in einem .NET-Framework-Stringausdruck, wie im Modifizierer angegeben, verwendet.

- **FormatiertesDatum**: Der Stringwert wird als Datum und Zeit interpretiert und ein formatierter String, der das Datum in Kurzform darstellt, wird gemäß den regionalen Einstellungen des Client-Computers zurückgegeben. Es wird kein Modifizierer benötigt.
- **FormatierteZeit**: Der Stringwert wird als Datum und Zeit interpretiert und ein formatierter String, der die Zeit in Kurzform darstellt, wird gemäß den regionalen Einstellungen des Client-Computers zurückgegeben. Es wird kein Modifizierer benötigt.
- **FormatierteBildURI**: Der Stringwert wird als Dateiname (ohne Erweiterung) in einer relativen URI zur PNG-Bilddatei im Ressourcenverzeichnis des Webservers verwendet.

Zusammenfassung

Das mit dem Produkt gelieferte Beispiel ist der beste Ausgangspunkt. Wie zuvor erwähnt, enthält das Verzeichnis ClientBin\Resources des Webservers eine Beispieldatei „Card.xaml“ und mehrere andere Dateien, auf die sie sich bezieht. Außerdem enthält der Ordner eine Textdatei „ExampleCustCardCustomSQL.txt“, die das SQL enthält, das dem Beispiel „Card.xaml“ entspricht. Dieser SQL-Text wird bei der Konfiguration des benutzerdefinierten SQL-Skripts im Dialogue Admin-Tool verwendet.

Das System bietet ein hohes Maß an Flexibilität. Häufig können Sie auswählen, ob eine komplexe Aufgabe im SQL-Skript oder im Konverter ausgeführt werden soll. Die mitgelieferte Datei „Card.xaml“ enthält Kommentare, die die Verwendung des Konverters beschreiben, und sie enthält auch Verwendungsbeispiele für unser DualBild-Steuerelement, das einen komfortablen Weg bietet, verschiedene Bilder anhand eines Booleschen Wertes anzuzeigen.

Ergebnisdaten-Integration

Da Interaction Optimizer Empfehlungen liefert und Antworten aufzeichnet, erstellt er einen für die Analyse nützlichen Verlauf der Kontakte. Sie können diese Ergebnisse mithilfe von Analyseanwendungen von Drittanbietern analysieren und dabei die Interaction Optimizer-Datenbank direkt nutzen.

Portrait Data Warehouse ist die Datenbank, welche die konsolidierte Ansicht der Anforderungs-, Behandlungs- und Antwortdaten der Marketingkampagne enthält. Dieser Datenspeicher wird auch verwendet, um Daten in ein Format zu aggregieren, das von den Kampagnenüberwachungsseiten in Portrait HQ leicht abgerufen werden kann. Der Zweck dieses Datenspeichers ist, dass er von SQL Server Reporting Services genutzt werden kann, um Berichte über die Kampagnen-Leistung zu generieren und sowohl eingehende als auch ausgehende Kampagnen-Behandlungs- und Antwortdaten in einzelne Kampagnenübersichten und Übersichten auf Aktivitätsebene zu konsolidieren.

Operationale Relationen

Die folgenden Relationen im konsolidierten Datenspeicher können als betriebliche Daten enthaltende Relationen angesehen werden, also die Daten, auf die von IO oder PD zugegriffen werden kann, um historische Informationen auf Kundenbasis abzurufen, wie z. B. die Anzahl der eingehenden Kontakte für einen bestimmten Kunden oder das letzte Datum, wann einem Kunden ein bestimmtes Angebot unterbreitet wurde. Diese Relationen werden auch herangezogen, um das Stern-Schema der Zusammenfassungsinformationen zu erstellen, das von Marketing HQ zur Anzeige der Details auf den Kampagnenüberwachungsseiten verwendet wird.

REQUEST_LOG

Diese Relation enthält die Details der eingehenden Anfragen, wie sie von IO erfasst wurden.

Anfordern	Beschreibung	Wert
RQL_REQUEST_ID	Primärschlüssel für diese Relation. Dies ist eine einfache Identitätsspalte.	BIGINT
RQL_SOURCE_ID	Die ID dieser Zeile in der Staging-Relation der IO-Datenbank. Sie wird verwendet, um diese Anfrage mit einer oder mehreren Aufforderungen in der Relation TREATMENT_OFFER_LOG zu verknüpfen.	BIGINT
RQL_SIM_REPLAY_ID	Wenn die Anfrage im Rahmen einer Simulationswiedergabe gestellt wurde, ist dies die ID der Wiedergabe; kann NULL sein.	INT
RQL_CUSTOMER_ID	Die ID des Kunden, der die Anfrage stellt.	(abhängig von der Installation)
RQL_AGENT_ID	Die ID des Vertreters, der die Anfrage bearbeitet.	NVARCHAR(40)
RQL_CHANNEL_ID	Die ID des CHANNEL, auf dem die Anfrage gestellt wurde; Verknüpfung zur Spalte CH_CHANNEL_ID der Kanal-Relation.	INT
RQL_APP_NAME	Der Name der Anwendung, die die Anfrage stellt; kann NULL sein.	NVARCHAR(50)
RQL_REASON_FOR_CALL	Zusätzliche Kontextinformationen zum Grund für die Anfrage; kann NULL sein.	NVARCHAR(50)
RQL_ENGAGEMENT_TYPE	Zusätzliche Kontextinformationen zum Typ der Zuordnung; kann NULL sein.	NVARCHAR(50)
RQL_LANGUAGE	Der Spracheinstellungscode für die aufrufende Anwendung, z. B. en_GB für UK-Englisch; kann NULL sein.	NVARCHAR(10)
RQL_TIMEZONE	Der Zeitzonencode für die Anfrage; kann NULL sein.	NVARCHAR(10)
RQL_REQUEST_TIMESTAMP	Der Zeitstempel, wann die Anfrage gestellt wurde, gemäß IO-System.	DATETIME
RQL_REGISTERED_TIMESTAMP	Der Zeitstempel, wann der Datensatz in diese Relation kopiert wurde (nur für systemexterne Verwendung).	DATETIME

TREATMENT_OFFER_LOG

Diese Relation wird zur Aufzeichnung der einzelnen Aufforderungen von IO verwendet (im Gegensatz zu tatsächlich für den Kunden durchgeführten Behandlungen, obwohl das bei direkten Kanälen dasselbe ist).

Behandlungsangebot	Beschreibung	Wert
TOL_ID	Der Primärschlüssel für die Relation; ein einfach inkrementierender IDENTITY-Wert.	BIGINT
TOL_DOMAIN_ID	Die Kennung der PD-Domäne; für Datensätze von IO mit -1 angegeben.	BIGINT
TOL_CUSTOMER_ID	Die ID des Kunden, der die Aufforderung/Behandlung empfängt.	(abhängig von der Installation)
TOL_AGENT_ID	Der Vertreter, der die Konversation auf einem indirekten Kanal bearbeitet.	NVARCHAR(40)
TOL_CONTEXT	Zusätzliche Kontextinformationen (nur PD).	NVARCHAR(128)
TOL_REQUEST_ID	Die ID der Anfrage, die zu dieser Nachricht/Behandlung aufforderte. HINWEIS: Dies verknüpft zur Spalte RQL_SOURCE_ID von REQUEST_LOG.	BIGINT
TOL_SRC_TREAT_ID	Die ID dieses Datensatzes in der IO Quell-Staging-Relation; dient zur Verknüpfung mit der Antwort in RESPONSE_LOG.	BIGINT
TOL_TREATMENT_ID	Die ID des Datensatzes der Relation TREATMENT; verknüpft mit Kanal und Vorhersageinformationen.	INT
TOL_INTERACTION_ID	Die Verknüpfung mit dem CAMPAIGN_ACTIVITY-Datensatz (verknüpft mit der Spalte CA_ACTIVITY_ID).	INT
TOL_OFFER_ID	Die ID des durch diese Aufforderung/Behandlung gemachten Angebots; kann NULL sein, wenn kein explizites Angebot gemacht wurde. Verknüpft mit der OFFER-Relation, Spalte OFR_OFFER_ID.	INT
TOL_MESSAGE_ID	Die Verknüpfung mit der ACM_MESSAGE_ID der ACTIVITY_CHANEL_MESSAGE, die durch diese Aufforderung/Behandlung präsentiert wird.	INT
TOL_TARGET_LIST_ID	Die ID der zur Auswahl von Teilnehmern bei dieser Aktivität verwendeten LIST. Kann NULL sein.	INT

Behandlungsangebot	Beschreibung	Wert
TOL_TREATMENT_BATCH_ID	Dient zur Identifizierung eines Behandlungsbatchs; wird von PD zur Identifizierung von Angebotsgruppen verwendet, die von derselben Behandlung gemacht wurden; wird in IO zur Identifizierung von Gruppen von Aufforderungen von derselben Anfrage verwendet.	BIGINT
TOL_PCM_CD_ID	Nur PD	INT
TOL_PCM_DP_ID	Nur PD	BIGINT
TOL_PCM_DOS_ID	Nur PD	INT
TOL_PCM_DBM_ID	Nur PD	INT
TOL_PCM_CM_ID	Nur PD	INT
TOL_IO_RANK	Nur IO Der Rang dieser Behandlung/Aufforderung, als sie präsentiert wurde.	INT
TOL_IO_SCORE	Nur IO Die mit dieser Aufforderung/Behandlung verknüpfte Bewertung.	FLOAT
TOL_IO_RECORDING_ID	Nur IO Die ID der Simulationsaufzeichnung, zu der dieser Eintrag gehören wird.	INT
TOL_SIM_REPLAY_ID	Nur IO Die Simulationswiedergabe, die diese Aufforderung generiert hat.	INT
TOL_TREATED_TIMESTAMP	Der Zeitstempel aus dem Quell-System, wann diese Behandlung/Aufforderung erfolgte.	DATETIME
TOL_REGISTERED_TIMESTAMP	Der Zeitstempel, wann der Datensatz in dieser Relation registriert wurde (nur für interne Verwendung).	DATETIME

SECONDARY_AUDIENCE_LOG

Diese Relation kann als eine Erweiterung der Relation TREATMENT_OFFER_LOG behandelt werden und dient der Aufzeichnung jeder Aufforderung für IO, die für sekundäre Zielgruppen bereitgestellt wurde.

Sekundäre Zielgruppe	Beschreibung	Wert
SAL_ID	Der Primärschlüssel für die Relation; ein einfach inkrementierender IDENTITY-Wert.	BIGINT

Sekundäre Zielgruppe	Beschreibung	Wert
SAL_CUSTOMER_ID	Die ID des Kunden, der die Aufforderung/Behandlung empfängt.	(abhängig von der Installation)
SAL_SRC_SECONDARYAUDIENCE_HISTORY_ID	Die ID dieses Datensatzes in der IO-Quell-Staging-Relation.	BIGINT
SAL_SRC_TREAT_HISTORY_ID	Die Behandlungsverlaufs-ID, die diesem Datensatz in der Staging-Relation des IO-Quellbehandlungsverlaufs entspricht.	BIGINT
SAL_OBJECT_NAME	Der Objektname der sekundären Zielgruppe, für welche die Aufforderung aufgezeichnet wurde (z. B. DecisionsDataSource.Accounts).	nvarchar (2000)
SAL_ID_NAME	Der Kennungsname der sekundären Zielgruppe, für welche die Aufforderung aufgezeichnet wurde (z. B. AccountNumber).	nvarchar (2000)
SAL_ID_VALUE	Der Kennungswert der sekundären Zielgruppe, für welche die Aufforderung aufgezeichnet wurde (z. B. die tatsächliche Kontonummer).	nvarchar(255)
SAL_TREATED_TIMESTAMP	Der Zeitstempel aus dem Quellsystem, wann diese Behandlung/Aufforderung für die sekundäre Zielgruppe erfolgte.	DATETIME
SAL_REGISTERED_TIMESTAMP	Der Zeitstempel, wann der Datensatz in dieser Relation registriert wurde (nur für interne Verwendung).	DATETIME

RESPONSE_LOG

Diese Relation wird zur Aufzeichnung der Details von Antworten auf für Kunden vorgenommenen Behandlungen verwendet. Bei einem indirekten System liefert diese Relation die tatsächlich ausgeführten Behandlungen (da die Behandlung aus der Tatsache gefolgert werden kann, dass eine Antwort vorliegt).

HINWEIS: Die Relation BEHAVIOR_LOG speist diese Relation und kann Antworten enthalten, die für keine Kampagnenaktivität gültig sind (weil sie Verhalten erfasst, die eine Antwort sein können oder nicht). Nur bestätigte Antworten auf Kampagnenaktivitäten werden im Antwortprotokoll gespeichert, und deshalb sollte nur diese Relation zur Abfrage des Antwortverlaufs herangezogen werden.

Antwort	Beschreibung	Wert
RL_ID	Primärschlüssel für die Relation; ein einfach inkrementierender IDENTITY-Wert.	BIGINT
RL_SRC_RESP_HIST_ID	Nur IO Die ID des Antwortverlauf-Datensatzes in der Staging-Relation in IO.	BIGINT

Antwort	Beschreibung	Wert
RL_TREATMENT_LOG_ID	ID des TREATMENT_LOG-Eintrags, auf den dies eine Antwort ist.	BIGINT
RL_BEHAVIOR_LOG_ID	Die ID des entsprechenden Eintrags der Verhaltensprotokoll-Relation.	BIGINT
RL_ACTIVITY_ID	Die Kampagnenaktivität, für die dies eine Antwort ist.	INT
RL_TREATMENT_ID	Die ID des TREATMENT-Relationseintrags, für den dies eine Antwort ist.	INT
RL_OFFER_ID	Die ID des OFFER-Relationseintrags, für den diese Antwort gilt.	INT
RL_MESSAGE_ID	Nur IO Die MESSAGE_ID-Verknüpfung zur ACTIVITY_CHANNEL_MESSAGE-Relation; wird zur Verknüpfung mit der Nachricht verwendet, auf die dies eine Antwort ist.	INT
RL_DOMAIN_ID	Nur PD Die Kennung der PD-Domäne; für IO mit -1 hartcodiert.	INT
RL_CUSTOMER_ID	Die Kennung des Kunden, der die Antwort tätigt.	(abhängig von der Installation)
RL_IO_AGENT_ID	Nur IO Die ID des Vertreters, für den die Antwort auf die Behandlung erfolgte.	NVARCHAR(50)
RL_CONTEXT	Nur PD Zusätzliche Kontextinformationen	NVARCHAR(128)
RL_PRODUCT_CODE	Ein Code zur Identifizierung des Produktangebots, auf das dies eine Antwort ist.	NVARCHAR(255)
RL_MESSAGE_TEMPLATE_ID	Nur PD Die ID der Nachrichtenvorlage, die bei der Behandlung verwendet wurde, auf die dies eine Antwort ist.	INT
RL_IO_RESPONE_NAME	Nur IO Der Name der vom Kunden getätigten Antwort.	NVARCHAR(50)

Antwort	Beschreibung	Wert
RL_RESPONSE_INDICATOR	Ein Wert zur Angabe des Antworttyps: 1 = Positiv 0 = Neutral -1 = Negativ	INT
RL_IS_SOFT_RESPONSE	Nur PD Ein Kennzeichen, das angibt, dass die Antwort gefolgert und nicht nachverfolgt wurde, z. B. wenn eine anonyme Kampagnenaktivitätsantwort erfasst wird.	BIT
RL_RESPONSE_COST	Die Kosten der Antwort.	MONEY
RL_RESPONSE_VALUE	Der Wert der Antwort.	MONEY
RL_VALUE_NET_OF_MARGN	Der Wert der Antwort als Nettogewinn.	MONEY
RL_TREATED_TIMESTAMP	Der Zeitstempel, wann die Behandlung stattfand.	DATETIME
RL_ACTED_TIMESTAMP	Der Zeitstempel der Antwort	DATETIME

Zusammenfassungsrelationen und Berichtsschema

Die übrigen Relationen in der Datenbank werden dazu verwendet, einfache Berichtserstellungsmöglichkeiten über SQL Server Reporting Services (SSRS) zu bieten. Die Zusammenfassungsrelationen für Kampagnen, Aktivitäten und Behandlungen werden durch den SSIS-Gesamtmengenprozess aktualisiert um sicherzustellen, dass sie die aktuellen Definitionen und Zusammenfassungen der Behandlung, Antworten und Kosten enthalten. Das Schema enthält auch zwei zentral „Fakt“-Relationen (eine für Behandlungen und eine andere für Antworten). In diesen Relationen werden die historischen laufenden Summen der Behandlungen, Antworten und Kosten für jede Kombination von Behandlung/Angebot/Kanal im Stundenintervall gespeichert.

TREATMENT_FACT

Die Behandlungsfaktenrelation enthält die laufenden Summen für Behandlungen/Angebote pro Kampagnenaktivität. Diese Relation verfolgt die laufenden Summen der Zähler und Kosten der Behandlung mit einer Stunden-Granularität (wie durch die Granularität der Relation TIME_DIMENSION definiert).

Behandlungsfakt	Beschreibung	Wert
TF_FACT_ID	Der Primärschlüssel für die Relation; eine einfach inkrementierende Identitätsspalte.	BIGINT
TF_TD_ID	Die Verknüpfung mit der Spalte TIME_DIMENSION.TD_ID, dient zur Identifizierung von	BIGINT

Behandlungsfakt	Beschreibung	Wert
	Stunde/Tag/Monat/Jahr, die durch diesen Datensatz repräsentiert werden.	
TF_DOMAIN_ID	Die PD-Domain (oder -1, falls aus IO), zu der die Kampagne gehört, für die diese Antworten aufgezeichnet wurden.	INT
TF_OFFER_ID	Die Verknüpfung mit der Spalte OFFER.OFR_ID zur Identifizierung des durch diese Behandlung gemachten Angebots, das durch diesen Datensatz repräsentiert wird.	INT
TF_TREATMENT_ID	Die Verknüpfung mit der Spalte TREATMENT_TR_ID zur Identifizierung der Behandlung, die durch diesen Datensatz repräsentiert wird.	INT
TF_ACTIVITY_ID	Die Verknüpfung mit der Spalte CAMPAIGN_ACTIVITY.CA_ACTIVITY_ID für die Details der Kampagnenaktivität.	INT
TF_TREATMENT_COUNT	Die aktuelle Summe (zur durch den Eintrag TIME_DIMENSION repräsentierten Zeit) der bisher gemachten Behandlungen. In IO ist dies der Zähler der Aufforderungen; bei PD ist dies der Zähler der Behandlungen, z.B. versendete E-Mails.	INT
TF_TREAT_OFFER_COUNT	Dieser Wert wird verwendet, wenn eine einzelne Behandlung mehrere Angebote machen kann, z.B. eine Mail von PD, die mehrere Angebote für einen einzelnen Kunden enthält. Dies ist der laufende Zähler der gemachten Angebote. In IO ist dieser Wert derselbe wie der Behandlungszähler (weil jede Behandlung aus einem einzelnen impliziten Angebot besteht). Wie beim Behandlungszähler ist dies die aktuelle Summe zu der durch den entsprechenden TIME_DIMENSION-Eintrag repräsentierten Zeit).	INT
TF_TREATMENT_COST	Die aktuellen Gesamtkosten für die Durchführung der Behandlungen.	MONEY
TF_PRODUCT_CODE	Der mit dem gemachten Angebot verknüpfte Produktcode (kann NULL sein).	NVARCHAR(255)

Behandlungsfakt	Beschreibung	Wert
TF_MESSAGE_TEMPLATE_ID	Nur PD Die ID der Nachrichtenvorlage in PD, die zur Übermittlung der Behandlung/des Angebots verwendet wird.	INT

RESPONSE_FACT

Die Antwortfaktenrelation enthält die entsprechenden Details für die Antworten, so wie die Behandlungsfaktenrelation sie für Behandlungen enthält.

Antwortfakt	Beschreibung	Wert
RF_FACT_ID	Der Primärschlüssel für die Relation; eine einfach inkrementierende Identitätsspalte.	BIGINT
RF_TD_ID	Die Verknüpfung mit der Spalte TIME_DIMENSION.TD_ID, dient zur Identifizierung von Stunde/Tag/Monat/Jahr, die durch diesen Datensatz repräsentiert werden.	BIGINT
RF_DOMAIN_ID	Die PD-Domain (oder -1, falls aus IO), zu der die Kampagne gehört, für die diese Antworten aufgezeichnet wurden.	INT
RF_OFFER_ID	Die Verknüpfung mit der Spalte OFFER.OFR_ID zur Identifizierung des Angebots, auf das diese Antworten erfolgten.	INT
RF_TREATMENT_ID	Die Verknüpfung mit der Spalte TREATMENT_TR_ID zur Identifizierung der Behandlung, auf welche die Antworten erfolgten.	INT
RF_ACTIVITY_ID	Die Verknüpfung mit der Spalte CAMPAIGN_ACTIVITY.CA_ACTIVITY_ID für die Details der Kampagnenaktivität.	INT
RF_RESPONSE_COUNT	Die aktuelle Summe (zur durch den Eintrag TIME_DIMENSION repräsentierten Zeit) der bisher erfolgten **positiven** Antworten.	INT
RF_RESPONSE_VALUE	Dies ist die laufende Summe der Werte aller bisher erfolgten Antworten (dies ist der Bruttowert).	MONEY
RF_VALUE_NET_OF_MARGIN	Wenn die Vorhersagedaten eine Umsatzspanne definieren, enthält dieses Feld den Nettogesamtwert dieser Spanne (vor Abzug der Kosten).	MONEY

Antwortfakt	Beschreibung	Wert
RF_RESPONSE_COST	Die aktuellen Gesamtkosten der Antwort (wo die Antwort Erfüllungskosten enthält).	MONEY
RF_RESPONSE_INDICATOR	Ein Wert zur Angabe, ob die Antworten positiv (1), neutral (0) oder negativ (-1) sind.	INT
RF_PRODUCT_CODE	Der mit dem gemachten Angebot/der Aktivität verknüpfte Produktcode, für den diese Antworten sind (kann NULL sein).	NVARCHAR(255)
RF_MESSAGE_TEMPLATE_ID	Nur PD Die ID der Nachrichtenvorlage in PD, die zur Übermittlung der Behandlung/des Angebots verwendet wird.	INT
RF_IS_SOFT_RESPONSE	Nur PD Dieses Kennzeichen wird verwendet um anzuzeigen, dass diese Antwroten gefolgert sind (wenn es sich um eine anonyme Kampagnenaktivität handelt und keine direkte Verknüpfung zwischen der Antwort und einer bestimmten Behandlung besteht).	BIT

TIME_DIMENSION

Diese Relation wird einfach dazu verwendet, die Zeitgranularität für die Relation der Fakteinträge zu definieren. Diese Relation enthält einen Eintrag für jede Stunde eines jeden Tages für den Zeitraum zwischen dem 01.01.2000 und dem Tag nach dem aktuellen Systemdatum (solange wie das SSIS-Paket zur Befüllung des Datenspeichers in geplanter Weise läuft). Die Spalten dieser Datenbankrelation sind selbsterklärend.

Zusammenfassungs-/Dimensionsrelationen

Die Relation TREATMENT steht auf der niedrigsten Granularitätsstufe, auf der Zusammenfassungen gespeichert werden. Diese Relation enthält die Gesamtzählung der Behandlungen (oder Aufforderungen bei eingehenden Behandlungstypen) und die Zählung der aufgezeichneten Antworten. Jede Behandlung ist für eine Kampagnenaktivität (Interaktion) und einen Kanal spezifisch, auch wenn mehrere Behandlungsdatensätze definiert sein können (falls eine Behandlung zu einer Aktivität hinzugefügt, entfernt und dann später wieder hinzugefügt wurde). Die Zusammenfassungsrelation speichert auch die Kosten und den Wert der Antwortzusammenfassungen, sofern für die Aktivität Vorhersageinformationen erstellt wurden. Die Zähler in dieser Relation werden inkrementell bei jeder Iteration des SSIS-Paktes, das diese Datenbank befüllt, aufsummiert. Die Relation TREATMENT verknüpft zu ihrer übergeordneten Aktivität über die Spalte TR_ACTIVITY_ID.

Die Relation CAMPAIGN_ACTIVITY enthält die Zusammenfassungszähler über alle möglichen Behandlungen (und deren jeweilige Antworten) für diese Aktivität (Interaktion). Die Relation wird inkrementell bei jedem Lauf des SSIS-Pakets aktualisiert, um sicherzustellen, dass die Relation die aktuellste Daten-

sicht auf die Aktivitätsleistung enthält. Die Spalte CA_CAMPAIGN_ID verknüpft zum Eintrag der übergeordneten CAMPAIGN-Relation (CAM_CAMPAIGN_ID).

Beispielabfragen

Abfrage zum Abruf der Details der Antworten pro Kampagnenaktivität.

HINWEIS: In IO spiegeln die Behandlungszähler die Anzahl der Aufforderungen wider. Echte Behandlungszähler können nur aus der Anzahl der Antworten geschlossen werden.

```
SELECT c.CAM_NAME as [Campaign Name],
a.CA_NAME as [Interaction Name],
ch.CH_NAME as [Channel],
t.TR_TOTAL_TREATMENTS as [Prompt Count],
t.TR_TOTAL_TREATMENTS -(t.TR_TOTAL_NEGATIVE_RESPONSES + t.TR_TOTAL_NEUTRAL_RE-
SPONSES + t.TR_TOTAL_POSITIVE_RESPONSES)
as [Not Presented Count],
t.TR_TOTAL_POSITIVE_RESPONSES as [Positive Responses],
t.TR_TOTAL_NEUTRAL_RESPONSES as [Neutral Responses],
t.TR_TOTAL_NEGATIVE_RESPONSES as [Negative Responses]
FROM [TREATMENT]t
INNERJOIN [CAMPAIGN_ACTIVITY] a ON t.TR_ACTIVITY_ID = a.CA_ACTIVITY_ID
INNERJOIN [CAMPAIGN] c ON a.CA_CAMPAIGN_ID = c.CAM_CAMPAIGN_ID
INNERJOIN [CHANNEL] ch ON t.TR_CHANNEL_ID = ch.CH_CHANNEL_ID
```

Um dieselbe Abfrage für eine Simulationswiedergabe auszuführen, muss folgendes SQL genutzt werden (Hinzufügen einer where-Klausel zur Auswahl einer bestimmten TOL_IO_SIM_REPLAY_ID):

```
                                    SELECT c.CAM_NAME as [Campaign Name],
                                    a.CA_NAME as [Interaction Name],
                                    ch.CH_NAME as [Channel],
                                    tl.TOL_IO_SIM_REPLAY_ID as [Replay ID],
                                    SUM(
                                    CASE
                                    WHEN r.RL_RESPONSE_INDICATOR ISNULLAND
tl.TOL_ID ISNOTNULLTHEN 1
                                    ELSE 0
                                    END)as [Not Presented Count],
                                    SUM(
                                    CASE
                                    WHEN RL_RESPONSE_INDICATOR = 1 THEN 1
                                    ELSE 0
                                    END)as [Positive Response],
                                    SUM(
                                    CASE
                                    WHEN RL_RESPONSE_INDICATOR = 0 THEN 1
                                    ELSE 0
                                    END)as [Neutral Response],
                                    SUM(
                                    CASE
                                    WHEN RL_RESPONSE_INDICATOR =-1 THEN 1
                                    ELSE 0
                                    END)as [Negative Response]
                                    FROM [TREATMENT] t
                                    LEFTOUTERJOIN [TREATMENT_OFFER_LOG] tl
ON tl.TOL_TREATMENT_ID = t.TR_TREATMENT_ID
                                LEFTOUTERJOIN [RESPONSE_LOG] r ON tl.TOL_ID
 = r.RL_TREATMENT_LOG_ID
                                    INNERJOIN [CAMPAIGN_ACTIVITY] a ON
```

```
t.TR_ACTIVITY_ID = a.CA_ACTIVITY_ID
                                    INNERJOIN [CAMPAIGN] c ON a.CA_CAMPAIGN_ID
 = c.CAM_CAMPAIGN_ID
                                     INNERJOIN [CHANNEL] ch ON t.TR_CHANNEL_ID
 = ch.CH_CHANNEL_ID
                                      GROUPBY c.CAM_NAME ,
                                      a.CA_NAME ,
                                      ch.CH_NAME,
                                      tl.TOL_IO_SIM_REPLAY_ID
```

Abfrage zum Abrufen der Zähler von Aufforderungen nach Kanal und Rang (nur IO verwendet Rang für Aufforderungen).

```
                                      SELECT c.CAM_NAME as [Campaign Name],
                                      a.CA_NAME as [Interaction Name],
                                      ch.CH_NAME as [Channel],
                                      SUM(
                                      CASE
                                      WHEN TOL_IO_RANK = 1 THEN 1
                                      ELSE 0
                                      END)as [Rank #1],
                                      SUM(
                                      CASE
                                      WHEN TOL_IO_RANK = 2 THEN 1
                                      ELSE 0
                                      END)as [Rank #2],
                                      SUM(
                                      CASE
                                      WHEN TOL_IO_RANK = 3 THEN 1
                                      ELSE 0
                                      END)as [Rank #3],
                                      SUM(
                                      CASE
                                      WHEN TOL_IO_RANK = 4 THEN 1
                                      ELSE 0
                                      END)as [Rank #4],
                                      SUM(
                                      CASE
                                      WHEN (TOL_IO_RANK <1 OR TOL_IO_RANK >
4)THEN 1
                                      ELSE 0
                                      END)as [Rank Other]

                                      FROM [TREATMENT] t
                                      LEFTOUTERJOIN [TREATMENT_OFFER_LOG] tol
 ON t.TR_TREATMENT_ID =tol.TOL_TREATMENT_ID
                                      INNERJOIN [CAMPAIGN_ACTIVITY] a ON
t.TR_ACTIVITY_ID = a.CA_ACTIVITY_ID
                                    INNERJOIN [CAMPAIGN] c ON a.CA_CAMPAIGN_ID
 = c.CAM_CAMPAIGN_ID
                                     INNERJOIN [CHANNEL] ch ON t.TR_CHANNEL_ID
 = ch.CH_CHANNEL_ID
                                      GROUPBY c.CAM_NAME ,
                                      a.CA_NAME ,
                                      ch.CH_NAME
```

Zum Filtern nach Simulationswiedergaben fügen Sie eine Klausel für ‘tol.TOL_IO_SIM_REPLAY_ID’ gleich NULL ein, um Simulationswiedergaben auszuschließen, oder für eine bestimmte Wiedergabe.

Reference Data Groups

Die Werte aus „configurable enumerations“ werden als Reference Data Group/RDI-Paare gespeichert. Diese beziehen sich auf Werte von Reference Data Group und Reference Data Item, die in den Relationen `amc_rd_ref_data_group` und `amc_rd_ref_data_item` aufgezählt werden.

Um einen RDI-Index zu einem konkreten Wert aufzulösen, gleichen Sie einfach seine Werte für Reference Data Group und RDI mit den Spalten `reference_data_group_id` und `reference_data_item_id` in der Relation `amc_rd_ref_data_item` ab.

Um die möglichen Werte für eine bestimmte Reference Data Group aufzulisten, listen Sie einfach alle Werte aus der Relation `amc_rd_ref_data_item` mit passender `reference_data_group_id` auf.

Um die Reference Data Group für einen bestimmten Type zu ermitteln, gleichen Sie ihren Reference-Data-Group-Wert mit der Spalte `reference_data_group_id` in der Relation `amc_rd_ref_data_group` ab.

Automatisierung

Dieser Auszug wird häufig als ein automatisierter, geplanter Datenextraktionsprozess eingerichtet. Einzelheiten, wie Sie dazu vorgehen, sind nicht Gegenstand dieser Dokumentation. Nachfolgend jedoch einige Hinweise und Tipps:

- Packen Sie Ihr SQL in ein SQL Server Integration Services-Paket, richten Sie dann mithilfe des SQL Server Agent Manager in SQL Server Management Studio automatsch geplante Jobläufe ein.
- Packen Sie Ihr SQL in Aufrufe an eine Batch-Engine wie SQLCMD, und erstellen Sie eine Zeitplanung mit Windows Scheduler oder einem ähnlichen Hilfsprogramm.
- Integrieren Sie den Extraktionsprozess in einen externen Analyse-Dateierzeugungsprozess (wie z. B. dem Quadstone System-Datenerstellungsbefehl `qsbuild`).

Protokollierung von Daten aus externen Tools

Wenn Interaction Optimizer zur Erfassung von Kampagnenreaktionsdaten verwendet wird, erfolgt die Protokollierung der Reaktionen mithilfe der Operation „Kampagnen-Reaktionen protokollieren“. Wenn Reaktionsdaten aus einem System protokolliert werden müssen, das sich außerhalb von Interaction Optimizer befindet, ist dies durch einen direkten Import in die Portrait Data Warehouse (PDW)-Datenbank möglich.

HQ-Fehlerbehebung

Fehler beim Authentifizieren bei Portrait HQ nach der Installation von SharePoint

Unter Windows Server 2003 stellen Sie möglicherweise nach der Installation von SharePoint fest, dass Sie sich nicht mehr bei Portrait HQ (nachdem Sie es installiert haben) oder anderen Webanwendungen authentifizieren können, welche die Windows-Authentifizierung verwenden, es sei denn, Sie geben beim Besuch der Site explizit Anmeldeinformationen des Kontos der lokalen Domäne an. Das ist ein IIS 6.0

Sicherheits-„Merkmal“. Um dieses Problem zu beheben, führen Sie die Schritte aus, die in *http://support.microsoft.com/?kbid=896861* beschrieben sind, wenn Sie die Windows-Authentifizierung in anderen Webanwendungen auf dem Server, wie beispielsweise Portrait Shared Services, zulassen möchten. Die einfachste der vielen hier beschriebenen Lösungen ist die Verwendung von *regedit*, um einen DWORD-Registry-Schlüssel „DisableLoopbackCheck“ mit dem Wert 1 unter *HKLM\System\CurrentControlSet\Control\Lsa* einzufügen.

Probleme mit den Sicherheitsanmeldeinformationen beim Ausführen von Portrait HQ

Abhängig von verschiedenen Konfigurationseinstellungen auf dem Servercomputer, dem Client-Browser und der Netzwerkinfrastruktur kann es vorkommen, dass ein Windows-Popup-Dialogfenster Sie nach einer Benutzer-ID und einem Kennwort fragt, wenn Sie versuchen, Portrait Miner zu starten. (Dieses Dialogfeld wird vom Internet Explorer angezeigt. Es ist nicht der Anmeldebildschirm von Portrait Miner.)

Für dieses Problem gibt es eine Reihe möglicher Ursachen. Sie sollten Folgendes in der angegebenen Reihenfolge überprüfen und alle erforderlichen Konfigurationsänderungen vornehmen.

- Läuft der Client (Browser) auf demselben Netz (Domäne) wie der Server? Wenn nicht: Besteht ein Vertrauensverhältnis zwischen diesen Domänen?
- Ist Internet Explorer für die Standard-Sicherheitseinstellungen zur Adressierung der lokalen Intranet-Zone konfiguriert? Wenn nicht: Versuchen Sie, diese Zone auf ihre Standardeinstellungen zurückzusetzen.
- Ist mit dem Portrait Shared Server ein Satz von Dienstprinzipalnamen im Active Directory verknüpft? Unter Umständen müssen SPNs für den Dienst erstellt werden. Anleitungen für die Einstellung von SPNs finden Sie weiter unten.

Fehlerbehebung bei einer „hängenden“ Anwendung beim Versuch der Anmeldung

Überprüfen Sie, ob die Benutzerprotokollierung über die notwendigen „Anmelde“-Berechtigungen verfügt.

Probleme beim Laden von Portrait Shared Services

Wenn `Http://<yourserver>:<port>/PortraitSharedService` nach der Installation nicht erreichbar ist, gehen Sie folgendermaßen vor:

1. Öffnen Sie web.config im Webverzeichnis
2. Blättern Sie zu `<"system.web">`
3. Fügen Sie unter dieser Zeile Folgendes ein: `<trust level="Full" originUrl="" />`

Erstellen von Dienstprinzipalnamen für Portrait Shared Services

Dienstprinzipalnamen werden im Active Directory mithilfe des Tools SETSPN erzeugt. Dieses Tool ist standardmäßig unter Windows Server 2008 verfügbar. **NB!** Änderungen an den Dienstprinzipalnamen (SPN) können die Windows-Authentifizierung für Portrait Dialogue-Webanwendungen beeinflussen. Wie Sie dies vermeiden, wird am Ende dieses Abschnitts beschrieben.

Stellen Sie nach der Installation des Support-Tools sicher, dass Sie auf dem Windows Server-Computer mit einem Konto mit Domänen-Administratorrechten angemeldet sind, und öffnen Sie dann einen Befehl Aufforderung. Führen Sie die beiden folgenden Befehle aus:

- `setspn -a http/<Servername> <Dienstkonto >`
- `setspn -a http/<vollqualifizierter Domänenname des Servers> < Dienstkonto>`

wobei Folgendes gilt:

- *Servername* ist der einfache (netbios) Name des Servers, wie beispielsweise „myserver“.
- *Dienstkonto* ist der Name des Dienstkontos, wie beispielsweise „mydomain\myaccount“.
- *Vollqualifizierter Domänenname des Servers* ist der vollständige Name des Servers, wie beispielsweise „mydomain\myaccount“.

Beispiele:

- `setspn -a http/myserver mydomain\myaccount`
- `setspn -a http/myserver.mydomain.mycompany.com mydomain\myaccount`

Zur Bestätigung, dass die Dienstprinzipalnamen erstellt wurden, führen Sie `setspn -l mydomain\myaccount` aus, und überprüfen Sie, ob die einfache und die vollqualifizierte Version des Servernamens in der Liste der mit dem Dienstkonto verknüpften Dienstprinzipalnamen stehen.

Beachten Sie, dass nur eine Instanz eines Dienstes auf einem Computer konfiguriert sein kann. Beispielsweise kann „HTTP/MyMachine“ mit dem „MyDomain\MyUser“ verknüpft sein. Das bedeutet, dass alle HTTP-(Web)-Sites/Dienste auf MyMachine unter MyUser account laufen müssen. Deshalb ist es nicht möglich, Webanwendungen unter App-Pools mit unterschiedlichen Entitäten auszuführen, wenn die Windows-Authentifizierung erforderlich ist. Das heißt, dass alle App-Pools auf einem einzelnen Domänenkonto ausgeführt werden müssen.

Nachdem Sie die oben genannten Schritte abgeschlossen haben, führen Sie die folgenden Schritte aus, um sicherzustellen, dass die Windows-Authentifizierung für Portrait HQ-Webanwendungen funktioniert:

1. Klicken Sie mit der rechten Maustaste auf **DefaultAppPool**.
2. Wählen Sie **Properties**.
3. Klicken Sie auf der Registerkarte **Identity** auf **Configurable**.
4. Fügen Sie die Anmeldeinformationen des Benutzers hinzu, der während der Erstellung der Dienstprinzipalnamen für Portrait Shared Services erstellt wurde.

Protokollierung

Weitere Informationen zur Aktivierung der Protokollierung in Portrait Shared Services oder Portrait HQ finden Sie in den Anweisungen im Abschnitt *Konfiguration*.

Deinstallieren von Dialogue Server nach der Installation von SharePoint und Portrait HQ

Wenn Dialogue Server nach der Installation von SharePoint und Portrait HQ deinstalliert werden muss, müssen unbedingt die folgenden Schritte ausgeführt werden. Der Grund dafür ist, dass der Benutzer, der den Prozess `Windows SharePoint Services Timer` ausführt, derselbe Benutzer ist, der das

Paket PD COM+ ausführt. Nach der Neuinstallation von Dialogue Server könnte dieser Benutzer von Active Directory wegen zu vieler fehlgeschlagener Authentifizierungsversuche gesperrt werden. Beachten Sie also Folgendes bei der Neuinstallation von Dialog Server auf einem Server, auf dem Portrait HQ und SharePoint installiert sind:

1. Vergewissern Sie sich, dass der oben genannte Dienst deaktiviert ist.
2. Installieren Sie Dialogue Server neu.
3. Starten Sie den Dienst neu.

Ändern das als Dienstkonto verwendete Konto nach der Installation von PSS

Um das Konto zu ändern, das nach der Installation von PSS als Service Account verwendet wird, gehen Sie folgendermaßen vor:

1. Ändern Sie das Konto, das als Identität des für PSS verwendeten Anwendungspools konfiguriert ist. Dazu wählen Sie in IIS im Anwendungspool `Advanced Settings`:
2. Führen Sie die unter „Post install mandatory configuration“ beschriebenen Schritte für das neue, als Dienstkonto zu verwendende Konto aus.
3. Führen Sie die unter „Creating Service Prinzipalnamen for the Portrait Shared Services“ beschriebenen Schritte für das neue, als Service Account zu verwendende Konto aus.

Erstellte Aufgaben werden nicht unter „Meine Ansicht“ in Portrait HQ angezeigt

Stellen Sie sicher, dass den Benutzern Aufgaben zugewiesen sind oder das Erstellen von Aufgaben in SharePoint richtig konfiguriert ist. Weitere Informationen finden Sie unter „Erstellen einer SharePoint-Site“ im:

- *Interaction Optimizer-Installationshandbuch* oder
- *Portrait Dialogue-Installationshandbuch*

Anzeige der Fehlermeldung „Fehler beim Abrufen der Aufgabenliste“ in Portrait HQ

1. Stellen Sie sicher, dass SharePoint entsprechend den Anweisungen im Interaction Optimizer-Installationshandbuch konfiguriert wurde.
2. Überprüfen Sie, ob die Endpunktadresse in der Datei `web.config` richtig ist:

```
<!-- Endpoint for outbound connection to SharePoint server's List Service
 (used for tasks, etc) -->
<endpoint address="http://vm-pss-sharep/portrait/_vti_bin/Lists.asmx"
binding="basicHttpBinding" bindingConfiguration="Sharepoint"
contract="ListsService.ListsSoap" name="ListsSoap"/>
```

3. Die in der Datei web.config angegebene URL darf nicht den Namen des Servers enthalten, auch wenn die URL funktioniert, wenn sie mit einem Webbrowser aufgerufen wird. Beispiel: Die Verwendung einer URL mit dem Wert `http://MyServer/MySharepointSite/portrait/_vti_bin/Lists.asmx` in der Datei „web.config“ funktioniert, wenn sie mit dem Browser angesprochen wird. Doch damit sie innerhalb von web.config gültig ist, muss der Name des Servers aus der URL entfernt werden: `http://MySharepointSite/portrait/_vti_bin/Lists.asmx`.

Kapitel

13

Hinweise zu Drittanbietern

In diesem Abschnitt:

- Hinweise zu Drittanbietern 348

Hinweise zu Drittanbietern

Die in diesem Handbuch genannten Namen von Firmen und Produkten, Marken, Logos und Zeichen sind möglicherweise Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer.

Dieses Produkt enthält:

- Software von OpenSSL Project für die Verwendung im OpenSSL Toolkit (http://www.openssl.org/). Die Lizenz für diese Software kann heruntergeladen werden unter: http://www.openssl.org/source/license.html. Copyright © 1998-2008 OpenSSL Project. Alle Rechte vorbehalten.
- Software von Eric Young (eay@cryptsoft.com) und Tim Hudson (tjh@cryptsoft.com). Copyright © 1995–1998 Eric Young (eay@cryptsoft.com). Alle Rechte vorbehalten.
- Software von Chad Z. Hower (Kudzu) und der Indy Pit Crew – http://www.IndyProject.org/. Copyright (c) 1993–2003, Chad Z. Hower (Kudzu) und die Indy Pit Crew. Alle Rechte vorbehalten. Die Software ist gemäß der BSD-Lizenz lizenziert. Sie kann unter http://www.indyproject.org/license/bsd.en.aspx heruntergeladen werden.
- Facebook_Csharp_SDK Version FB: 6.0.20 ist gemäß der Apache-Lizenz Version 2, Januar 2004 lizenziert. Die Lizenz kann heruntergeladen werden unter: http://www.apache.org/licenses/LICENSE-2.0.html. Der Quellcode für diese Software ist unter http://csharpsdk.org/ verfügbar.
- Twitterizer2version FB: 2.4.1 ist gemäß der BSD Simplified-Lizenz lizenziert. Die Lizenz kann heruntergeladen werden unter: http://www.twitterizer.net/license. Der Quellcode für diese Software ist unter http://www.twitterizer.net verfügbar.

Index

1 59, 63, 166
    1-Gruppenausdrücke 59
    n-Gruppenausdrücke 59, 63, 166

A

Accept 222
AcceptAll 223
AcceptBySQL 223
AcceptBySQLRaw 223
ActiveXObject 128, 129, 251, 255, 263, 301
Activities 63, 274
Activity API 251
ActivityDataXML 251
ActivityDesc 243, 274
ActivityID 251
ActivityTypeDescName 166, 251
ActivityTypeID 243
ActivityTypeName 166, 243, 251, 274
AddField 143
AddMessage 217
AddParam 147, 165
AddRow 143
Admin 12, 42, 58, 255, 263, 289, 301
ADO 196
ADO Dataset 196, 251, 263
ADO-Datensatz 241
ADO.NET 251, 263
Aggragated 305
Aktivitäten 59, 61
Aktivitätstypen 19
Allgemeine Eigenschaften 42, 44, 48
Allgemeine Verwaltung 32
AND 61
Anmelden 32
Anmeldung 12, 250
Anrufstatustypen 16
AnswerDateTime 166
AnswerFormDataXML 289, 305
AnswerFormID 289
AnswerForms 47, 63
AnswerFormURL 58, 63
AnswerNo 63, 138
Antwortformulare 59
Anwendungssystem 32
Anzeigeintervalle 19
API 250
ApplicationSystem 301
ArchiveRef 138, 166
Archivierter Bericht 293
Array 123, 126, 196
AssignedToUserName 305
AssignToUserName 305
Asynchronously 255
Aufgabe 19
Aufgaben 19
Aufgabenfeld 19
Ausdruck 15, 36, 58
Ausdrucksanalyse 58
Ausdrucksoperatoren 61
Ausgabekanäle 20
Ausgangskanal-Plug-in 153
Ausgangskanal-Plug-In 129
Auswahl-Designer 58
Auswahldesigner 39, 61, 63

B

Base64Binary 274
BaseMessageID 243, 274
Benutzer 25, 31
Benutzergruppen 25
Bericht 293
Berichtsarchiv 293
Berichtsformat 293
Berichtsvorlage 293
Beschreibung 263
BoolAnswer 61, 63
Branch 148, 190
Branch Plug-in 178
BranchInfo 126, 131, 143, 148, 166, 178, 196, 222, 223

C

CallComplete 305
CallingUser 128
CallStatus 305
CanSetDataFields 128, 155
CCProjectID 305
Cellular 129, 242
ChannelType 61
ChannelTypeName 166, 251
CharDelimiter 166, 178, 274
Clear 223
ClientLocalPath 274
ClientMachine 274
CollectionName 196
ColNo 63
ColumnNo 138
COM 23, 122, 138, 148, 155, 188, 210, 221, 227, 250, 251, 255, 263, 271, 274, 289, 301, 305
COM-Implementierungen 138
Comments 138, 166
Connect 129, 251, 255, 263, 301
    DialogServer 251, 255, 263, 301
ConnectionName 196, 223, 271
Container 166
ContentBinary 211, 274
ContentText 211, 274
Context 166, 211, 223, 251, 255, 263, 274, 289
ContextExpression 223, 263, 274
ContextIncluded 223
Control Parameters 274
ControlParamDefs 129, 165, 221
ControlParams 166, 211, 243, 274
ControlParamValue 211, 220, 243
Count 153, 178, 187, 217
CreateAnswerForm 166
CreateMergeFile 166, 178, 274
CreateMessage 166, 178
CreateMessageBundle 178
CreateSingleMergeFile 274
CreateSingleWordMergeFile 274
CurrentCallStatus 305
CustDomainID 126, 131, 143, 148, 178, 217, 243, 251, 255, 263, 274, 289
Customer API 305
Customer View 38, 39, 40, 42, 43, 46, 48, 49, 50, 251, 274
Customer Web Access 40, 50, 63
CUSTOMER_MESSAGE 274
Customer-API 263
CustomerAPI 263, 301
CustomerContainer 128, 155
CustomerDataXML 128, 263
CustomerID 128, 166, 211, 223, 251, 255, 263, 274, 289
CustomerMessageID 274

D

DataFields 128, 155, 166, 178, 243, 255, 263, 274, 305
DataGroupName 166
Datasbase 196, 271
Dataset 48, 50, 196, 251, 263, 271
Datatype 143, 186
Datenbankinstanzen 12
Datenfelder 42
Datengruppe 51
Datengruppen 43
Datenquelleneigenschaften 45
Datensatz 43
Datentyp 15
Datentypen 51
DatePart 63
Datetime 51, 63, 131, 196, 271
DateValue 63
DBMS 12
DefaultValue 147, 165
Definitionen der Ausdruckssyntax 59
Delete 138
DeleteActivity 251
DeleteAnswerForm 289
DeleteAnswerForms 289
DeleteAnswerFormsUNC 289
DeleteCustomerList 298
DeleteCustomerMessage 274
DeleteMessage 274
DeletePsrCustomerList 298
DeletePsrSelection 298
DeleteRow 123, 188
DeleteSelection 298
Description 166, 186, 251
DetailDataEof 166
DialogAPI 255
DialogID 255
Dialogue Admin 12, 36, 37, 40, 42, 58, 143, 166, 178, 196, 220, 223, 241, 242, 243, 250, 251, 255, 263, 271, 274, 301, 305
Dialogue API 251, 271
Dialogue Database 12, 16, 19, 23, 32, 178, 222, 223, 243, 255, 289, 301
Dialogue Host 301
Dialogue Manager Service 196
Dialogue Server 12, 16, 22, 23, 32, 36, 40, 46, 47, 49, 59, 63, 122, 123, 126, 128, 129, 131, 138, 145, 148,

153, 155, 166, 178, 188, 196, 220, 221, 227, 243, 250, 251, 255, 263, 271, 274, 301
Dialogue Server API 271, 301
Dialogue Server-API 16, 19, 250
Dialogue Server-Host 12
Dialogue-API 255
Diffgram 251, 263
Direction 166, 251, 274
DisplayName 242
DLG_PARTICIPANT 223
Documents 61
Dokumente 59
Domänen 36, 37, 39
DynamicList 126, 131, 143, 148

E

E-Mails 15, 43
Eigenschaften von Kundendomänen 42
Einrichtung 15
Emails 128, 129, 178
Empfängergruppe 223
EncodeDateTime 63
Epression 263
Ereignistypen 16
ErrorMessage 153
ErrorMessages 153
ErrorText 129
ErrorUNC 289
EvalExprString 166
EvaluateExpression 263
EvaluateExpressionString 263
EventTypeName 166, 263
ExcludeBySQL 223
ExcludeBySQLRaw 223
ExecuteBranch 126, 131, 143, 148, 166, 178, 196, 222, 223
ExecuteOperation 255
ExecuteOperationAsync 255
ExecutePlugin 128, 196, 271
ExecuteSQL 271
ExecuteSQL_JS 271
ExecuteSQLScript 271
ExecuteStoredProc 271
ExecuteStoredProc_JS 271
ExportAnswerForms 289
ExportAnswerFormsUNC 289
Exportieren 112
ExportUNC 289
Expression 166, 263
Expression functions 63

F

Felder 38, 47
Feldname 38
Fenster „Gruppeneigenschaften“ 37
Fenster „Suchquellen definieren“ 51
FetchNextToCall 305
FetchToCall 305
FieldByName 187
FieldBySourceName 187
Fieldname 143, 166, 186, 187, 263
FieldsAvailable 128, 155
FieldValue 166
FileName 166, 178
FilterExpression 223, 255, 263, 274, 305
FilterOnRecall 305
FilterSQL 126
FirstDetailData 166
FirstMember 153, 178
FollowUpDateTime 166, 251
FollowUpUserName 166, 251
FromGroup 145
FromNumber 129, 165
FUIntervalBegin 251
FUIntervalBeginSet 251
FUIntervalEnd 251
FUIntervalEndSet 251
Fuzzy-Nachrichten 32

G

Generic API 128, 271
Generic Plug-ins 128
GetActivity 251
GetActivitySchema 251
GetActivityTypes 251
GetAnswerForm 166, 289
GetAnswerFormByID 289
GetCallStatusTypes 305
GetCategories 263
GetCCParticipantCount 305
GetCCParticipantDetails 305
GetCCParticipants 305
GetCCProjects 305
GetChannelTypes 255
GetControlParamDefs 129, 165, 220, 221
GetCustDomains 263
GetCustomerAsXML 166
GetCustomerContexts 255
GetCustomerCount 263
GetCustomerLists 298
GetCustomers 263
GetCustomersAsXML 178

GetCustomerSchema 263
GetCustomerSelection 263
GetCustomerSelectionCount 263
GetCustomSqlFieldInfo 274
GetDataset 271
GetDialogDetails 255
GetDialogHistory 255
GetDialogs 255
GetDynamicList 126, 131, 143, 148
GetExpressionFunctions 263
GetFuzzyMessages 274
GetInstances 301
GetLookupDatarow 263
GetLookupDataset 263
GetMasterTemplate 274
GetMessageDetails 274
GetMessagesLockedByUser 274
GetMessageTypes 274
GetNextFilename 243
GetParamDefs 126, 131, 143, 145, 147, 148
GetParticipantCount 255
GetParticipantHistory 255
GetParticipants 255
GetParticipantSchema 255
GetParticipantSelection 255
GetParticipantSelectionCount 255
GetQuestionnaire 289
GetQuestionnaires 289
GetSelections 263, 298
GetSequenceID 196, 271
GetServerVersion 301
GetSingleCustomer 263
GetSingleCustomers 263
GetSingleParticipant 255
GetSQLDef 196
GetStatistics 305
GetSystemUserInfos 301
GetTableTextAnswer 138
GetTaskFUOptions 251
GetTasksByUser 251
GetTaskSummaryByUser 251
GetTaskViewIntervals 251
GetTemplate 274
GetTemplateDetails 274
GetTemplates 274
GetTemplateSchema 274
GetTextAnswer 138
GetUnmergedMessage 178
GetUserSessionInfo 301
GetWorkGroups 251
GroupID 255
Gruppe 43
Gruppentypen 15

H

Hauptdatengruppe 37
Hauptgruppenausdrücke 59
HTTP POST 250

I

IBM DB2 12
IfString 63
IMHActivityAPI 251
IMHAnswerForm interface 138
IMHBranchDynamicList-Schnittstelle 143
IMHBranchInfo-Schnittstelle 145
IMHBranchParamDefs-Schnittstelle 147
IMHBranchPlugin-Schnittstelle 148
IMHCCCallStatusList-Schnittstelle 151
IMHCCProject-Schnittstelle 151
IMHControlParamDefs-Schnittstelle 165
IMHCreateMessagePlugin-Schnittstelle 155
IMHCustomer-Schnittstelle 166
IMHCustomerAPI 263
IMHCustomerContainer interface 178
IMHCustomPluginServices-Schnittstelle 185
IMHDataField interface 186
IMHDataFields-Schnittstelle 187
IMHDataGroupPlugin interface 188
IMHDialog-Schnittstelle 190
IMHDialogAPI 255
IMHDialogGroup-Schnittstelle 193
IMHDialogOperation interface 195
IMHDialogServerServices 196
IMHDialogServerServices interface 196
IMHGenericPlugin interface 210
IMHGenricAPI 271
IMHMessage interface 211
IMHMessageAPI 274
IMHMessageBundle interface 217
IMHMessageContainer interface 153
IMHOutputChannelInfo-Schnittstelle 220
IMHOutputChannelPlugin-Schnittstelle 221
IMHParticipant-Schnittstelle 222
IMHParticipantContainer-Schnittstelle 223
IMHPlugin-Schnittstelle 227
IMHQuestionnaire-Schnittstelle 231
IMHQuestionnaireAPI 289
IMHSelection-Schnittstelle 240
IMHSelectionAPI 298
IMHSQLDef-Interface 241
IMHSystemAPI 301
IMHSystemUser interface 242
IMHTelemarketingAPI 305
IMHUnmergedMessage interface 243

ImportAnswerForms 289
ImportAnswerFormsUNC 289
Importieren 113
Inactivate 222
InactivateAll 223
InactivateParticipant 255
IncludeContent 274
IncludeContext 126
IncludeDone 251
IncludeUndone 251
Index 187, 217
Inhalt 50
Inhaltstypeigenschaften 50
InitializePlugin 227
InsertActivity 251
InsertParticipant 255
InsertRow 123, 188
InstanceName 251, 255, 263, 301
InternalID 178
Interneteigenschaften 50
IsContentBinary 243
IsInSelection 63
IsMergeParam 165, 220
IUnknown 227

J

JavaScript 40, 49
JScript 23, 122, 126, 128, 251, 255, 263, 271, 301

K

Kanaltypen 19
Kategorien 15
KeyFieldname 263
KeyValue 263
Komplementäre Umgebung 114
Komplementäres Importieren 113
Kundendaten-Plug-Ins 123
Kundendatenbank 36, 38
Kundendomänen 12, 15, 36, 37, 39, 40, 42
Kundendomänen-Editor 37
Kundenprofil 50

L

Length 63
ListAsString 151
LockedByUserName 305
LockMessage 274
LogDebugMessage 12, 196
Login 242, 271, 301
LoginEx 301
Logout 271, 301
LookupSourceName 263
LowerCase 63

M

MainDataGroupName 178
MarkAsSent 274
MarkTaskAsDone 251
MaxRows 271
Mehr Informationen zu Domänen 39
MemberEof 178
MergeControlParams 166
Message 155, 196
Message API 274
Message Plug-ins 178
MESSAGE_LOG 274
MessageID 274
MessageLogID 211
MessageName 243, 274
MessageText 129
MessageTypeID 243
MessageTypeName 178, 243, 274
MessageUNC 211
MH Dialog Manager-Dienst 12
MH Send Messages Service 178
MH Test 251, 255, 263
Mh_copy_from_remote_group 131
Mh_copy_to_remote_group 131
MH-Nachrichteversanddienst 12
MH-Test 301
MHGenericParam 271
Microsoft ADO.NET 251, 263, 271, 289, 305
Microsoft ASP.NET 40
Microsoft COM 210, 250
Microsoft Word 39
MS OLE DB 12
MS SQL-Server 12
MS Word Mail Merge 274
MS Word Merge 128, 166, 178
MTS 23

N

Nachrichten-Plug-Ins 128
Nachrichtentypen 22
Nachverfolgung 19
Name/Kennwort 12
NewsLetter 63, 166, 222
NewValue 123, 186
NextDetailData 166
NextMember 178
Nicht-komplementäres Importieren 113

NOT 61, 289
Now 63
NumberOf 59, 63

O

ObjectName 185
Objektsicherheit 29
ODBC 23
OLD DB 23
OldValue 123, 186
OLE DB-Verbindung 23
OnePerCustomer 128
OperationID 255
Operationstypen 16
OR 61
Oracle 12, 196, 271
OUT 166, 251, 274
Outbox 178, 243, 274
Output Channels 129, 221
OutputChannelID 243
OutputChannelInfo 221
OutputMessage 128
OutputMessages 128, 129, 221

P

Param 129, 211, 243
ParamAsFloat 196
ParamAsInteger 196
ParamAsString 196
ParamDefs 126, 131, 143, 147, 148
Parameter 12, 42
Parametersammlungen 32
ParamName 126, 131, 143, 145, 148, 196, 211, 220, 243
ParamType 147
ParamValue 145
ParticipantID 222, 255, 274, 305
Participants 126, 131, 143, 148, 166, 178, 196, 222, 223
Password 301
PercentCompleted 145, 220
Plug In-Repository 23
Plug-in 128
Plug-In-API 12, 122, 138
Plug-In-Repository 46
Plug-Ins 46
PluginDefID 243
PluginName 196, 271
PopulateFromSQL 143
PopulateFromXML 143
Portrait Shared Repository 298
PostActivity 166, 251
PostEvent 166, 263
PostMessages 178
PostSingleDialogMessage 274
PostSingleMessage 274
PostTask 166, 251
Process Monitor 12, 196
ProcessedCount 153
ProduceMessage 166
ProduceMessages 128, 155, 178
ProducePostMessages 274
ProducePostSingleDialogMessage 274
ProducePostSingleMessage 274
ProduceSingleDialogMessage 274
ProduceSingleMessage 128, 155, 274
ProduceSingleTestMessage 274
ProduceTestMessages 274
ProduceTestMessagesFromSelection 274
PSR 298
PSR Selection 298

Q

QuesionnaireID 63
Questionnaire API 289
QuestionnaireID 63, 166, 289
QuestionNo 63, 138
QuoteChar 166, 178, 274

R

RemoveCategory 166, 263
RemoveCategoryValue 166, 263
ReportProgressStatus 145, 220
Ressourcen 22
RowNo 63, 138

S

Safearray 210
SavePsrCustomerList 298
SavePsrCustomerListAsynchronous 298
SavePsrSelection 298
SaveSelection 298
SaveSqlSelection 298
SaveTemplate 274
ScanDateTime 166
Schnittstellen 138
Schritt 36, 37, 38, 39
Score 166, 263
Scramble 63
Sekundäre Datenbanken 23
Select/divide 151

SelectByExpression 223
SelectBySelection 223
SelectBySQL 223
SelectBySQLRaw 223
SelectCategories 131, 143
Selection 298
Selection API 298
SelectionID 63, 223, 255, 263
SelectSingle 223
SendMessages 129, 221
SequenceName 196, 271
Server-API 250
Server-Hosts 12
ServerSession 250, 251, 255, 263, 271, 274, 289, 301, 305
SessionKey 251, 255, 263, 301
SetCategory 166, 263
SetCategoryScore 166, 263
SetCategoryValue 166, 263
SetCheckAnswer 138
SetDataFields 128, 155
SetRadioAnswer 138
SetStatusError 153
SetStatusOK 153
SetTableBoolAnswer 138
SetTableTextAnswer 138
SetTaskReadFlag 251
SetTextAnswer 138
Sicherheit 25, 31
Simple Category 131, 143, 147
SMTP 20, 23, 122
SOAP 250, 251, 255, 263, 271, 274, 289, 301, 305
SourceFieldName 186, 187
SourceUNC 289
SourceXML 289
SQL 36, 37, 38
SQL Repository 36, 223, 271, 289
SQL-Hauptgruppe 36
SQL-Repository 22, 51
SQLConnectionID 243
SQLConnectionObject 196
SQLExecute 196, 271
SQLExecuteRaw 196
SQLExecuteStoredProc 196
SQLName 143, 196, 223, 271, 289
SQLOpen 196
SQLOpenRaw 196
SQLRetrieveValue 196
SQLRetrieveValueRaw 196
SQLs 12, 22, 23, 39, 43, 45, 46, 48, 51, 63, 123, 126, 131, 143, 196, 223, 241, 243, 250, 271, 289
StatusText 145, 220
Statustypen 16
StoredProcName 196, 271
SuccessUNC 289
SuccessXML 289
Sucheigenschaften 48
Sucheinstellungen 50
Suchquellen 51
Suchschlüssel 48
Sum 63
System API 251, 301
System-API 250, 263
Systemgruppen 47
SystemUserName 63, 251

T

Tabellenwartung 32
TableBoolAnswer 63
TableTextAnswer 63
Task Organizer 251
TASK_FU_OPTION 251
TASK_VIEW_INTERVAL 251
TaskID 251
TaskWorkGroup 166, 251
TaskWorkGroupID 166, 251
Technische Hilfe 15
Telemarketing 40, 50, 305
Telemarketing API 305
Telemarketing-Web 50
TemplateUNC 243
Testen 39
    Domäne 39
Testing 250
TextAnswer 63
TimePart 63
Today 63
ToGroup 145
ToString 63

U

UnlockCCParticipant 305
UnlockMessage 274
UnmergedMessage 128, 155, 166, 178, 217
UnsubscribeURL 63
Unsubsribe-page 63
Updatable 49
UpdateActivity 251
UpdateAnswerForm 289
UpdateDataset 271
UpdateMessage 274
UpdateRow 123, 188
UpdateSingleCustomer 263
UpdateUserSettings 301
UpperCase 63

URLs 50, 58, 63
containing 63
opening 63
scramble 63
UseOutbox 178, 243
UseQuote 166, 178, 274
UserName 129, 242, 301
UsesSQL 243

V

Validierungseigenschaften 49
Verbinden 23
Dialogue Database 23
Verzweigung 131, 145, 193, 195
Verzweigungs-Plug-In 126
Verzweigungsparametertypen 131
Visual Dialogue 16, 29, 39, 40, 42, 58, 131, 143, 145, 147, 148, 223, 243, 251, 255, 274
Task Organizer 251
Von-Gruppe 223

W

Webdienstanwendung 250
Webdienste 23, 250
Webeigenschaften 43, 46
WebProfileURL 63
Webumgebung 40
Word Mail Merge 274

X

XML diffgram 251, 263, 271, 289, 305
XSL Transform 128